Analyse factorielle des correspondances pour l'indexation et la recherche d'information dans une grande base de données d'images

Khang-Nguyen Pham

▶ **To cite this version:**

Khang-Nguyen Pham. Analyse factorielle des correspondances pour l'indexation et la recherche d'information dans une grande base de données d'images. Interface homme-machine [cs.HC]. Université Rennes 1, 2009. Français. NNT : . tel-00532574

HAL Id: tel-00532574
https://theses.hal.science/tel-00532574

Submitted on 5 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre : 3970 ANNÉE 2009

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MATISSE

présentée par

Nguyen-Khang PHAM

préparée à l'unité de recherche UMR 6074 IRISA
Institut de recherche en informatique et systèmes aléatoires
Composante universitaire : IFSIC

Analyse factorielle des correspondances pour l'indexation et la recherche d'information dans une grande base de données d'images

Thèse soutenue à IRISA
le 6 novembre 2009

devant le jury composé de :

Israël-César LERMAN
Professeur émérite à l'Université de Rennes 1 / président

Djamel Abdelkader ZIGHED
Professeur à l'Université Lumière Lyon 2 / rapporteur

Georges QUÉNOT
Chargé de Recherche CNRS / rapporteur

Annie MORIN
Maître de Conférences à l'Université de Rennes 1 / directrice de thèse

Patrick GROS
Directeur de Recherche INRIA / co-directeur de thèse

Quyet-Thang LÊ
Maître de Conférences à l'Université de Cantho / co-directeur de thèse

Remerciements

Ces travaux ont été effectués en collaboration entre l'Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), au sein de l'équipe TEXMEX et l'Université de Cantho, au sein de la Faculté des technologies de l'information. La thèse est financée par une bourse de formation à la recherche de l'Agence Universitaire de la Francophonie (AUF).

J'aimerais remercier l'ensemble des membres du jury pour avoir accepté d'y participer, Israël-César LERMAN, professeur émérite de l'université de Rennes 1, qui me fait l'honneur de présider ce jury. Je tiens également à remercier Djamel Abdelkader ZIGHED, professeur de l'Université Lumière Lyon 2, et Georges QUÉNOT, chargé de recherche CNRS, d'avoir bien voulu accepter la charge de rapporteur.

Je souhaiterais remercier tout particulièrement Annie MORIN, Patrick GROS et Quyet-Thang LÊ pour la qualité de leur encadrement, la pertinence de leurs conseils ainsi que l'investissement formidable dont ils ont fait preuve tout au long de ces trois années.

Je remercie François POULET de m'avoir recommandé à Annie MORIN, pour ses relectures attentives et de m'avoir donné des conseils utiles. Je suis extrêmement redevable à Thanh-Nghi DO, pour avoir guidé mes premiers pas en tant que doctorant, pour de longues discussions révélatrices.

Je tiens à remercier aussi Michel KERBAOL. Mon outil de visualisation interactif CAViz est inspiré de ses idées développées dans son logiciel Bi-Qnomis. Les sympathiques discussions au café sur la parallélisation sur GPU avec Van-Hoa NGUYEN m'ont été particulièrement profitables.

Je voudrais remercier tous mes collègues de l'équipe TEXMEX pour leur accueil et les moments de détente partagés. Un merci tout particulier à Stacy PAYNE pour son aide sur la correction de l'anglais, à Loïc LESAGE pour les démarches administratives, à Pierre TIRILLY pour les discussions utiles et à Sébastien CAMPION pour l'intégration du moteur de recherche d'images par l'AFC à la plate-forme ISEC.

Table des matières

Introduction générale

Le problème de l'indexation et de la recherche de textes existe depuis longtemps. Il est connu des bibliothécaires et des documentalistes qui doivent gérer leurs ouvrages. Il existe des catalogues et des systèmes de classification qui permettent de résoudre le problème. Cependant, compte-tenu de l'explosion du nombre de documents, les catalogues doivent être mis à jour et la recherche efficace d'information devient de plus en plus difficile. Depuis quelques années, on a développé des méthodes automatiques d'indexation de bases de données textuelles et de recherche d'information.

Avec le développement du numérique, le nombre d'images stockées dans les bases de données a beaucoup augmenté. L'indexation des images et la recherche d'information dans les bases d'images sont plus compliquées que dans le cas de documents textuels. Quand il s'agit d'organiser des images, l'homme fait souvent mieux que les machines tant que la taille de la base n'est pas trop grande. Des méthodes d'indexation déjà utilisées en analyse de données textuelles (ADT) ont été proposées comme le *tf*idf* (term frequency-inverse document frequency) [SB88], le PLSA (Probabilistic Latent Semantic Analysis) [Hof99a, Hof99b] et le LDA (Latent Dirichlet Allocation) [BNJ03]. Ces méthodes nécessitent l'utilisation de nouvelles caractéristiques : les *mots visuels* qui permettent de répondre à cette demande.

L'objectif de cette thèse est l'indexation et la recherche d'informations dans une grande base de données d'images à l'aide de méthodes d'analyse de données et plus précisément de l'analyse factorielle des correspondances (AFC) [Ben73]. L'AFC est très utilisée en ADT et travaille sur des tableaux croisant mots et documents. Pour transférer les résultats de l'ADT aux images, on utilise des mots visuels, les documents étant les images en l'occurrence.

Description par chapitre

Les deux premiers chapitres de la thèse sont consacrés respectivement à de brefs rappels sur les principes de la recherche d'images par le contenu où nous expliquons la formation des mots visuels et décrivons les méthodes inspirées de l'ADT comme LSA, PLSA et LDA. Nous présentons ensuite des métriques possibles d'évaluation, puis aux méthodes d'indexation et de recherche d'images dans une base de données : indexation multidimensionnelle, recherche séquentielle accélérée et recherche approximative.

Le chapitre 3 décrit les principes de l'AFC et son adaptation pour l'analyse et la

recherche d'images. Nous définissons les mots visuels qui seront utilisés ultérieurement en faisant un clustering sur les descripteurs locaux.

Nous proposons une utilisation astucieuse des indicateurs de l'AFC pour accélérer la recherche : l'AFC permet la projection simultanée des images et des mots visuels dans un espace de dimension réduite. On peut mesurer la qualité d'un point projeté dans un espace réduit (sur un axe ou sur un plan), on peut interpréter un axe en calculant la contribution des points à l'inertie de cet axe. Nous terminons ce chapitre par des expérimentations sur trois bases d'images et nous comparons notre méthode à des méthodes plus classiques comme *tf*idf* en ADT [SZ03b] et PLSA [LS07].

Le chapitre 4 propose des solutions pour le passage à l'échelle. L'AFC est une méthode basée sur une décomposition en valeurs singulières. Nous proposons d'abord une AFC incrémentale pour traiter de grands tableaux de données, puis la parallélisation de l'algorithme précédent sur GPU. Nous envisageons aussi la parallélisation de l'algorithme de recherche approximative proposée dans le chapitre précédent sur GPU.

Quelques expérimentations sur de grandes bases de données démontrent l'intérêt des améliorations apportées par la parallélisation à l'AFC. Dans la seconde partie de ce chapitre, nous associons l'AFC à d'autres méthodes comme la Mesure de Dissimilarité Contextuelle (MDC) [JHS07] ou la forêt aléatoire construite à partir d'arbres obliques [DLPP09] pour améliorer le qualité de la recherche d'images.

Le chapitre 5 est consacré à l'outil de visualisation CAViz que nous avons développé pour accompagner les traitements des chapitres précédents. En effet, on commence avec une grande base d'images et l'AFC génère énormément de résultats qu'il faut compiler. La visualisation des résultats est importante pour la lecture des résultats et leur interprétation.

Enfin, dans la conclusion, nous présentons une synthèse des travaux effectués et quelques perspectives liées à cette étude. Notre contribution originale est développée dans les chapitres 3, 4 et 5 et a fait l'objet de publications dans des conférences nationales [PM08, PMGL09b] et internationales [PMG08e, PMG08b, PMG08f, PMG08a, PMGL08, PMG09], des journaux [PMG08c, PMG08d] et un ouvrage consacré à la fouille de données complexes [PMGL09a].

Chapitre 1

Principes de la recherche d'images par le contenu

1.1 Introduction

Ce chapitre est consacré aux principes de fonctionnement de la recherche d'images par le contenu (RIC) : on recherche, dans une base d'images, les images les plus proches d'une image de requête en n'utilisant que les informations visuelles extraites automatiquement de ces images. C'est une tâche non triviale pour deux raisons [SWS+00] :

- *Sensorielle.* Le *fossé sensoriel* est le fossé séparant l'objet réel de sa représentation numérique, sous forme d'image ou de description de cette image par exemple.
- *Sémantique.* Le *fossé sémantique* traduit la différence entre l'information extraite du contenu visuel des images et l'interprétation de ce contenu par l'utilisateur dans un certain contexte.

Dans un système RIC typique (figure 1.1), le contenu visuel des images de la base est extrait et décrit par ce qu'on appelle des *signatures d'images*. Les *signatures* des images de la base constituent une *base de signatures*. Pour rechercher des images, l'utilisateur fournit une image exemple (appelée la *requête*). Le système représente la requête par sa *signature*. Les mesures de similarités/dissimilarités entre la signature de la requête et celle de *toutes les images* de la base sont calculées et comparées. Le résultat est le plus souvent présenté sous forme d'une liste des images de similarité descendante. Comme toutes les images de la base doivent être examinées pour retrouver des images similaires à la requête, le coût devient prohibitif lorsque la taille de la base augmente. Pour remédier à ce problème, la plupart des systèmes RIC utilisent un *schéma d'indexation* qui fournit une méthode de recherche très efficace. L'idée principale est de ne pas parcourir toute la base mais d'examiner une seule petite partie de la base. Les techniques d'indexation seront présentées dans le chapitre 2. Certains systèmes récents ont intégré le *contrôle de pertinence* de l'utilisateur pour modifier le processus de recherche afin de générer des résultats plus significatifs. Il s'agit d'une technique qui permet de poursuivre une recherche d'information progressive avec l'interaction de l'utilisateur : l'utilisateur soumet une requête. Le système lui présente un ensemble de documents retrouvés. L'uti-

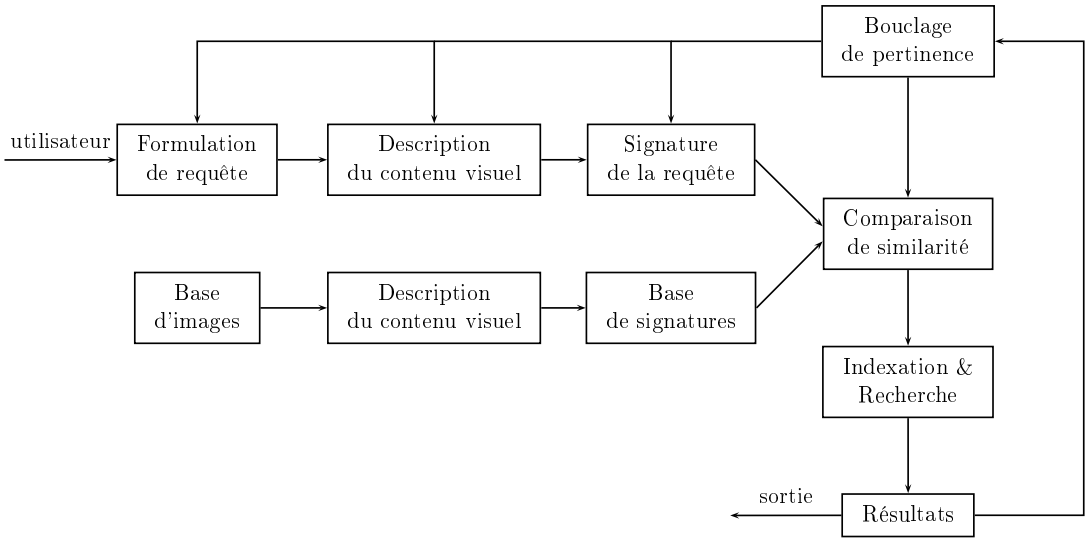


FIGURE 1.1 – Diagramme pour un système de recherche d'images par le contenu typique.

lisateur indique ensuite les documents qui sont pertinents pour son besoin. Le système peut utiliser cette information soit quantitativement (retourne plus de documents similaires aux documents pertinents) soit qualitativement (retourne documents similaires aux documents pertinents avant d'autres documents).

Ainsi, par la nature de ses tâches, un système RIC tente de résoudre deux problèmes : (i) comment décrire mathématiquement le contenu visuel d'une image, et (ii) comment évaluer la similarité entre deux images en utilisant leur description abstraite. Les solutions au premier problème sont présentées dans la section 1.2 qui traite la description du contenu visuel de l'image. Le second problème est détaillé dans la section 1.3. Nous décrivons ensuite brièvement les méthodes inspirées de l'analyse des données textuelles avant l'interaction homme - machine et les métriques d'évaluation.

1.2 Description du contenu visuel

La description du contenu visuel des images est une étape essentielle dans un système de recherche d'images par le contenu. Cette étape a pour but de fournir une représentation du contenu de l'image (appelée également la signature de l'image) qui permet des recherches efficaces. Il ne s'agit pas de coder toute l'information de l'image comme dans le cas de compression mais de se concentrer sur les informations pertinentes dans un objectif de recherche. La question qui se pose ici est la suivante : quelles *caractéristiques* doivent être extraites et comment décrire des images afin de pouvoir effectuer une recherche efficace ? La recherche est ici décrite comme une spécification des conditions invariantes minimales qui modélise l'intention de l'utilisateur, et qui est développée pour réduire le *fossé sensoriel* dû aux distortions accidentelles, aux occultations, à l'influence

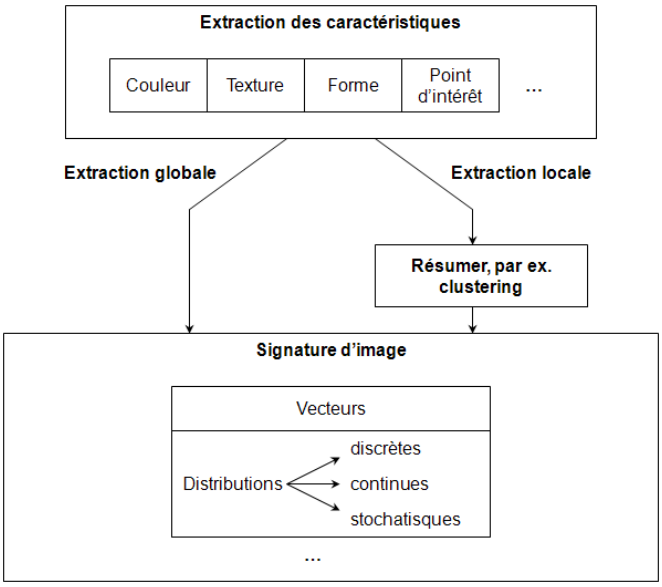


FIGURE 1.2 – Vue d'ensemble de la formulation mathématique des signatures d'images.

du fond, etc. Nous commençons par la définition de *caractéristiques visuelles* suivie par la description de certains types de caractéristiques couramment utilisées. Nous présentons ensuite la construction des signatures visuelles en se basant sur les caractéristiques extraites dans l'étape précédente. On peut avoir une vue d'ensemble de la construction des signatures d'images sur la figure 1.2.

1.2.1 Extraction des caractéristiques visuelles

Il n'y a pas une définition universelle ou exacte de ce qui constitue une *caractéristique*. La définition exacte dépend souvent du problème ou du type d'application. De ce fait, nous adoptons la définition suivante des caractéristiques :

Définition 1.1 (Caractéristique visuelle) – *Une **caractéristique visuelle** (ou **caractéristique** pour faire bref) d'une image est définie comme une abstraction des informations visuelles de l'image qui sont pertinentes pour des tâches de calcul reliées à une certaine application (par ex. classification d'images, recherche d'images).*

Les caractéristiques sont extraites, soit globalement sur une image entière, soit localement sur un petit groupe de pixels (une région). Le résultat d'une étape d'extraction de caractéristiques (globales ou locales) est appelé *descripteur de caractéristique*.

Définition 1.2 (Descripteur de caractéristiques) – *Nous appelons la description mathématique d'une image ou une région locale de l'image, après une étape d'extraction de caractéristiques, son **descripteur de caractéristiques** (ou **descripteur** pour plus de simplicité).*

Les descripteurs se présentent souvent sous forme d'un vecteur dans un espace vectoriel, $\mathbb{R}^D$, appelé *l'espace de caractéristiques*.

Dans le cas d'une extraction globale, on récupère un seul descripteur par image tandis qu'une description locale permet d'obtenir pour une image un ensemble de descripteurs locaux.

Nous présentons dans cette section certains types de caractéristiques les plus couramment utilisées pour calculer les descripteurs : **la couleur**, **la texture**, **la forme, les points d'intérêt** et **les relations spatiales**.

La couleur

Dès le début de la recherche d'images par le contenu (RIC), on a exploré les caractéristiques de couleurs et plus exactement les espaces de couleurs qui permettent de s'approcher au mieux du fonctionnement du système visuel humain. Bien que la plupart des images soient dans l'espace de couleur RGB, cet espace est rarement utilisé pour l'indexation et la recherche d'images car il ne correspond pas très bien à la perception de la couleur. D'autres espaces comme HSV (Hue, Saturation, Value), CIE L*a*b* ou CIE L*u*v*(proposés par la Commission Internationale de l'Éclairage) sont mieux par rapport à la perception humaine (c-à-d. les différences dans l'espace de couleur sont similaires à celles entre les couleurs que les humains perçoivent) et sont donc plus utilisés.

Les *moments de couleurs* ont été utilisés dans plusieurs systèmes de recherche d'images comme QBIC [FSN+95], en particulier quand l'image ne contient qu'un seul objet. Ils se sont avérés pertinents et efficaces pour la représentation des distributions de couleurs des images [SO95]. Comme seuls 9 (3 moments pour chaque composante de couleur) nombres sont utilisés pour décrire le contenu de chaque image, la représentation par les moments de couleurs est très compacte par rapport à d'autres caractéristiques de couleurs. En raison de cette compacité, le pouvoir de discrimination des moments de couleurs s'abaisse.

Proposés la première fois dans [SB91], les *histogrammes de couleurs* sont devenus des descripteurs principaux pour le contenu d'images et ont été utilisés dans QBIC [FSN+95], VisualSEEK [SC96] et Pictoseek [GS00]. Ils sont faciles à calculer et sont eux aussi efficaces pour de petites bases. Cependant, comme un histogramme de couleurs calcule globalement la fréquence des couleurs d'une image, des images très différentes peuvent donc avoir des distributions similaires. Ce problème devient très crucial surtout quand la taille de la base augmente. Pour remédier à ce problème, Pass *et al.* a proposé des *histogrammes joints* [PZ99]. Ces dernières améliorent le pouvoir de discrimination des histogrammes de couleurs en combinant plusieurs types de caractéristiques comme couleur, densité de contour (edge density), texture (texturedness), etc. Par ailleurs, les histogrammes de couleurs ne prennent pas en considération les informations spatiales. En fait, les pixels ayant de même couleur ne sont généralement pas similaires car ils peuvent représentent des coins, des contours ou des régions uniformes. D'après cette observation, Pass *et al.* a proposé encore une façon d'incorporer des informations spatiales dans les histogrammes de couleurs, les *vecteurs de couleurs cohérentes* (Color Coherence Vector) [PZ96]. Les pixels sont classifiés en deux classes : *cohérente* (s'ils se trouvent

dans une région de couleur uniforme) ou *incohérente* (sinon). On calcule ensuite, pour chaque classe de pixels, un histogramme de couleurs. Bien que les vecteurs de couleurs cohérentes fournissent de meilleurs résultats que les histogrammes de couleurs grâce aux informations spatiales incorporées, il n'y a cependant pas de distinction pour les pixels dans une même classe. Par ailleurs, la détermination de la classe pour les pixels nécessite quelques paramètres qui ne sont pas toujours les mêmes pour des images différentes.

La technique des *corrélogrammes de couleurs* [HKM+99] est une autre approche pour incorporer des informations spatiales dans les histogrammes de couleurs. Ils permettent de caractériser non seulement les distributions des pixels mais aussi la corrélation spatiale des paires de couleurs. Comparés aux histogrammes de couleurs et aux vecteurs de couleurs cohérentes, les corrélogrammes de couleurs fournissent les meilleurs résultats, mais leur complexité de calcul est plus coûteuse.

Les *histogrammes de couleurs pondérées* (weighted color histograms) [BBV01] améliorent les histogrammes de couleurs et les vecteurs de couleurs cohérentes en utilisant une pondération adaptative sur la contribution de chaque pixel. La pondération se base sur une mesure locale de la non-uniformité de couleurs, calculée dans un voisinage du pixel. Les histogrammes de couleurs pondérées font un compromis entre la robustesse et la complexité de calcul des descripteurs basés sur les couleurs.

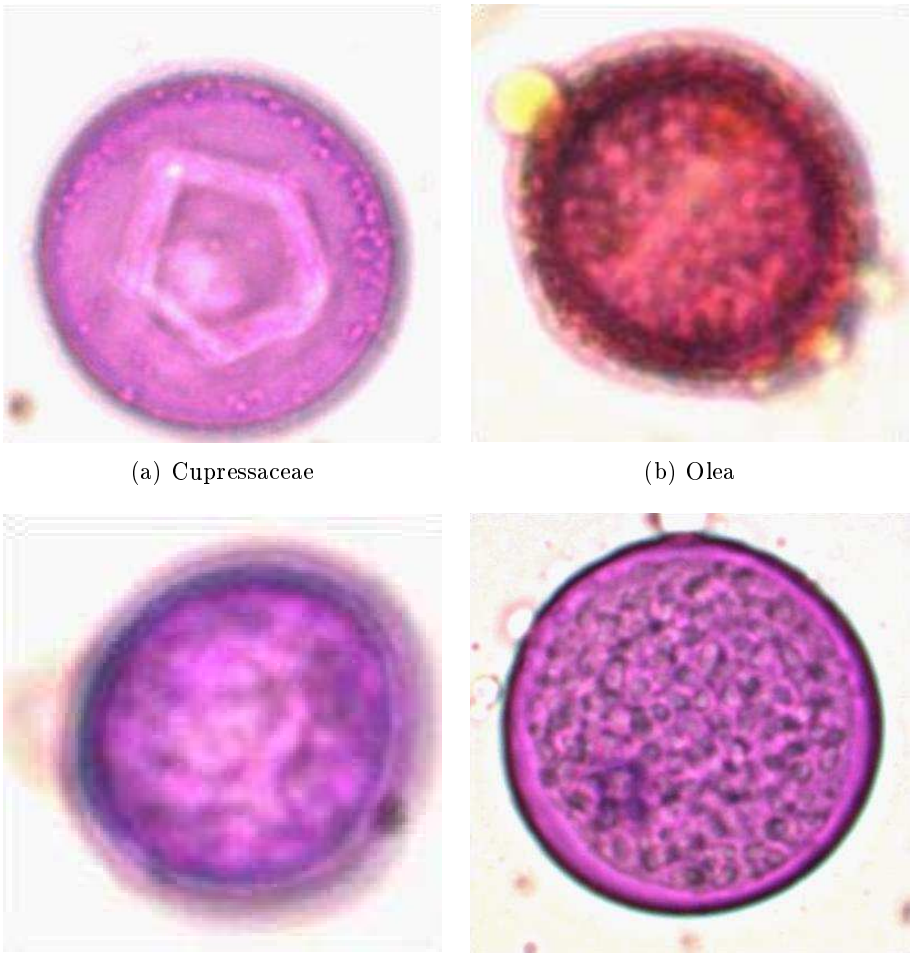
La texture

La texture est une autre propriété importante de l'image. Les caractéristiques de texture permettent de capturer la granularité et les motifs répétitifs des surfaces dans l'image. Par exemple, les pelouses, les murs de briques, les pétales de fleurs sont différents en terme de texture. Le premier travail notable sur le sujet est celui sur les descripteurs d'Haralick pour la classification automatique des roches [HS73]. La texture est un paramètre important dans des domaines comme l'imagerie aérienne ou l'imagerie médicale [MM98].

La représentation de la texture se divise en deux catégories dépendant du domaine d'application, reconnaissance de formes, vision par ordinateur ou infographie [Har79] : *structurelle* et *statistique*. Les méthodes structurelles, y compris les *opérateurs morphologiques* et les *graphes de contiguïté*, décrivent les textures en identifiant des primitives structurelles et leurs règles de placement. Elles sont très efficaces pour représenter des textures régulières. En revanche les méthodes statistiques caractérisent les textures par la distribution statistique de l'intensité d'image.

Les *descripteurs de Tamura* [TMY78] sont utilisés dans QBIC [FSN+95] et Photobook [PPS96]. Par contre, dans [MP90, JF91, Uns95, MM96, LWW00, DV02], les caractéristiques de texture sont calculées à partir de ces coefficients comme la *transformée en ondelettes*, la *transformée en cosinus* et les *filtres de Gabor*, etc. Les descripteurs basés sur champs aléatoires Markoviens sont aussi utilisés dans [MJ92, PS00].

La plupart des descripteurs de textures précédents font l'hypothèse que les images sont capturées sous une même condition. Ils ne sont pas invariants aux transformations affines (par ex. rotation, changement d'échelle). Si on cherche des caractéristiques invariantes aux transformations affines et photométriques [SZ01], on peut consul-



(a) Cupressaceae (b) Olea

(c) Parietaria (d) Poaceae

FIGURE 1.3 – Images des pollens dans l'application de reconnaissance des pollens (source : `http://www-sop.inria.fr/orion/ASTHMA`)

ter [Low04, MS04, BTVG06] qui utilisent la détection des *points d'intérêt* dans l'image.

La forme

La forme est un attribut important des objets ou des régions segmentées dans l'image. Comparées aux caractéristiques de couleur et texture, les caractéristiques de forme sont habituellement extraites après une étape de segmentation d'images en objets ou régions. Puisqu'il est jusqu'à présent difficile de faire une segmentation exacte et robuste et que la notion d'objet ou de région pertinents n'est pas clairement définie, l'utilisation des caractéristiques de formes se limite aux applications spéciales où on dispose d'objets ou de régions (Figure 1.3).

On a deux types de méthodes : les *méthodes basées sur les contours* (y compris formes

rectilinéaires [Jag91], approximation polygonale [ACH+91], et descripteurs de Fourier [PF86, ASBH90, KSP95]) et les *approches basées sur les régions* (moments statistiques [Hu62, YA94]).

Parmi les développements récents, on trouve la représentation des formes utilisant l'évolution des courbes discrètes pour simplifier des contours [LL00], la mise en correspondance des formes (shape matching), appelé *shape context* robuste à certaines transformations géométriques [BMP02], une approche par le contour représentant des formes par des séquences de segments concaves et convexes [PDM02]. Dans ce cas, la similarité entre formes est calculée par un algorithme de programmation dynamique. L'utilisation de l'amplitude et de la phase des coefficients de Fourier permet aussi de décrire les formes [BCP05]. Dans ce cas, la distance *dynamic time warping*, au lieu de la distance euclidienne, permet de faire une mise en correspondance exacte en présence d'un décalage (limité) de phase.

Les relations spatiales

Les régions et objets ayant des propriétés de couleur et texture similaires peuvent se distinguer facilement par l'introduction de contraintes spatiales. Par exemple, une région du ciel bleu et une région d'océan peuvent avoir deux histogrammes de couleurs similaires mais leurs localisations spatiales sont différentes. Ainsi, la localisation spatiale des régions (ou objets) ou la relation entre des régions multiples dans l'image est très utile pour la recherche d'images.

La représentation des relations spatiales la plus utilisée est le *2D string* proposé par Chang *et al.* [CSY87] basée sur la théorie de projections symboliques. Ses variantes et extensions sont le *2D G-string* [CJL88], le *2D C-string* [LH90], le *2D B-string* [LYC92] et le *2D Be-string* [Wan03].

Par ailleurs, d'autres modèles capturent les relations spatiales entre caractéristiques locales comme les points d'intérêt et représentent les catégories d'objets ou les classes visuelles par une combinaison de descripteurs locaux et de leurs distributions spatiales. L'introduction de connaissances *a priori* sur les relations spatiales entre parties locales conduit soit à des modèles avec des parties totalement indépendantes (sans relation spatiale comme le cas de sac-de-caractéristiques où chaque caractéristique représente une région), soit à des modèles avec des parties totalement connectées (modèle de constellation [FPZ03]), soit à des modèles intermédiaires (topologie d'étoile [FPZ05], structure hiérarchique [BT05] et structure où les parties dépendent spatialement de leurs voisins [BT05]). Enfin, les informations spatiales peuvent être intégrées dans une étape de post-traitement pour améliorer les résultats [SZ03b, Low04, SZ06].

1.2.2 Description locale des images

Pour obtenir des descripteurs locaux à partir d'une image, on commence par extraire des régions. La façon la plus simple est d'utiliser une *partition* qui découpe l'image en rectangles de même taille. Une telle partition simple ne génère pas des régions perceptuellement significatives mais c'est une manière simple d'obtenir des caractéristiques

globales de l'image avec une résolution plus fine. Nous présentons ci-dessous les deux approches les plus utilisées pour localiser les régions d'intérêt dans l'image : l'une fournit des régions qui se chevauchent (détection de points d'intérêt) et l'autre segmente l'image en régions sans intersection (segmentation d'image). Le mieux serait certainement de pouvoir segmenter l'image en objets (par ex. voiture, cheval, ballon) mais cela reste au delà des possibilités techniques actuelles.

Détection de points d'intérêt

Les *points d'intérêt* traditionnellement utilisés pour la *stéréo vision* sont utilisés aussi dans la recherche d'images. Ils sont déterminés de manière telle qu'un point trouvé dans une image sera aussi trouvé dans une autre image qui diffère légèrement de la première. La signification de tels points spéciaux est due à leur représentation compacte des régions importantes de l'image qui conduit à une indexation efficace, et à leur pouvoir discriminant surtout dans la recherche d'objets.

Un des premiers travaux sur le sujet [SM97] utilise un détecteur de Harris [HS88] pour localiser des points d'intérêt invariants à la rotation. Dans [DSH00], on montre que les descripteurs ne peuvent pas être invariants au changement d'échelle si les points d'intérêt extraits ne sont pas eux mêmes invariants au changement d'échelle. Par conséquent, plusieurs détecteurs ont été proposés pour obtenir l'invariance au changement d'échelle des points d'intérêt [Lin98, Low99, MS01, Low04]. La sélection automatique de l'échelle est effectuée en choisissant les *extrema* d'une fonction de l'échelle (par ex. laplacien normalisé, différence de gaussiennes). Un autre détecteur proposé dans [TSL+01] pour extraire des *points saillants* (*salient points*) explore les coefficients d'une transformation en ondelettes aux différentes échelles. Pour obtenir l'invariance aux transformations affines, Mikolajczyk *et al.* ont proposé un *détecteur de Harris adapté aux transformations affines* et un algorithme itératif pour la détection des points d'intérêt invariants au changement d'échelle et aux transformations affines [MS02, MS04].

Une discussion sur les avantages et les inconvénients des types de points d'intérêt couleurs utilisés dans la recherche d'images se trouve dans [GB02] tandis que des évaluations comparatives de divers détecteurs de points d'intérêt sont réalisées dans [SMB98, SMB00, MTS+05].

Caractérisation des points d'intérêt

Pour utiliser les points d'intérêt il faut alors caractériser la région autour de ces points. La caractérisation d'un point d'intérêt est calculée, à une échelle choisie, sur la région autour de ce point. Différents descripteurs ont été proposés dans la littérature : *Shape context* [BMP02], *Steerable filters* [FA91], *Scale Invariant Feature Transform* (SIFT) [Low04], PCA-SIFT [KS04], *Gradient Location and Orientation Histogram* (GLOH) [MS05]. Une étude comparative sur l'évaluation de la performance des descripteurs est présentée dans [MS05]. Parmi des descripteurs, le descripteur SIFT est le plus utilisé [SZ03b, WAC+04, SRE+05, BZM06, LS07].

Segmentation d'images

L'approche par segmentation de régions est la plus utilisée pour le paradigme de requêtes partielles. La segmentation des images est une question centrale en traitement d'images. Pour acquérir une représentation basée sur les régions, il faut segmenter l'image. L'avantage de la segmentation en régions pour la recherche d'images par le contenu est que le partitionnement en régions se rapproche plus d'une détection d'objets de nature sémantique. La fiabilité de cette description est cruciale pour la caractérisation des formes dans l'image. La plupart de techniques de segmentation d'images peuvent être classées grossièrement en deux catégories : les *approches basées sur les contours* et les *approches basées sur les régions*. Fondamentalement, les premières approches se basent sur la discontinuité et ont tendance à partitionner une image par la détection des points isolés, des lignes et des bords. Les régions sont déduites de leurs contours. Des méthodes basées sur une telle approche comprennent les filtrages locaux comme le détecteur de contours de Canny [Can86], ou les modèles de minimisation d'énergie comme les contours actifs (snake model) [KWT88] et les modèles de ballons [Coh91]. Les algorithmes de la seconde catégorie explorent l'homogénéité de certaines caractéristiques (intensité, couleur, texture, etc.) y compris le seuillage [SSWC88], le clustering [Pap92], la croissance de région [RJ97], la segmentation par division [HS79] et la segmentation par fusion [HEMK98]. Les deux approches ont leurs avantages et inconvénients. Pour améliorer les résultats de la segmentation, on peut combiner les deux approches [Ron94, ZY96, CD99, GDTG00]. Une des avancées les plus importantes dans le domaine de segmentation est l'approche basée sur les graphes. Le problème de segmentation devient un problème de partition de graphes où l'ensemble des sommets est composé des pixels, et le poids d'un arc joignant deux sommets (pixels) représente une certaine mesure de dissimilarité non-négative perceptible entre deux pixels. Une segmentation est une partition de l'ensemble des sommets en composantes telle que chaque composante (ou région) correspond à une composante connectée dans le graphe. Le problème de partitionnement d'un graphe est connu comme un problème de type NP-complet. Certaines approches approximatives ont été proposées comme le critère de coupures normalisées [SM00] ou la partition basée sur les prédicats mesurant la vraisemblance des contours entre deux régions [FH04]. Les techniques complètes pour l'indexation et la recherche d'images par le contenu basées sur la segmentation de régions se trouvent dans [CTB+99].

1.2.3 Formation des signatures d'images

La description mathématique des images c'est à dire leur *signature*[1] est le cœur des systèmes RIC. Les signatures d'une image sont construites à partir de ses caractéristiques visuelles et sont de deux types, des *vecteurs de caractéristiques* (feature vectors)

1. Dans la littérature, les deux termes *signature* et *descripteur* sont utilisés de manière interchangeable. Par convention, nous utilisons le terme *signature* pour la description d'une image entière tandis que le terme *descripteur* est utilisé pour indiquer la description d'une certaine partie de l'image (cf. extraction locale). Dans la cas d'extraction globale, la *signature* d'une image est son *descripteur* calculé à partir des caractéristiques de toute l'image.

et des *distributions des caractéristiques* (cf. Figure 1.2). Comme nous allons le présenter en détail par la suite, les histogrammes et les signatures basées sur les régions peuvent être considérées comme des *ensembles de vecteurs pondérés*. Quand la somme des poids est égale à 1, ces ensembles sont équivalents aux *distributions discrètes* (discrète au sens où les supports sont finis).

L'extraction globale décrit souvent les images par des *vecteurs de caractéristiques* tandis que l'extraction locale d'une image produit un *ensemble de descripteurs locaux* que l'on considère comme une distribution de descripteurs locaux. Chaque descripteur local est un vecteur qui caractérise le voisinage d'un pixel particulier (point d'intérêt) ou une région segmentée. Une utilisation directe de l'ensemble des descripteurs consiste à représenter une image par un ensemble de ses descripteurs locaux. Avec cette représentation, la recherche d'une image requête dans la base d'images nécessite une suite de recherches individuelles des descripteurs locaux de l'image requête ; les résultats sont fusionnés par une technique de vote [SM97, MGS98, Low99, TG99, MS01, AG01, SZ03a, BAG03, MS04, Low04].

Nous présentons maintenant la construction des signatures d'images basées sur le résumé des descripteurs locaux et la relation entre le résumé des descripteurs en histogramme et la signature basée sur les régions. Soient des descripteurs locaux, notés par $s_{i,j} \in \mathbb{R}^D, 1 \leq i \leq M, 1 \leq j \leq n_i$ où M est le nombre d'images et n_i est le nombre de descripteurs locaux de l'image i. Pour construire un histogramme basique, l'espace $\mathbb{R}^D$ est divisé en un nombre fixe de *cellules* et le pourcentage des $s_{i,j}$ qui se trouvent dans chaque cellule est calculé. Supposons qu'il y a k cellules, un histogramme sera traité comme un vecteur de k dimensions $[f_1\ f_2\ \ldots\ f_k]$ où f_l est la fréquence de la cellule l. C'est exactement une quantification vectorielle des descripteurs locaux. Les cellules sont traités comme des variables symboliques. Dans [SZ03b, WAC+04, SZ06, LS07, JHS07], les cellules sont définis comme des « *mots visuels* » qui sont analogues aux mots textuels dans les textes. Ainsi, les techniques de traitement de textes peuvent être appliquées sur les images où ces dernières sont représentées par des *sacs-de-mots-visuels* (modélisés par un *modèle vectoriel* avec une pondération ou plus simplement, un *modèle d'histogramme*). Un avantage de cette approche est que les images sont représentées par des vecteurs simples de même nombre de dimensions comme dans le cas d'extraction globale. Une telle représentation facilite les tâches ultérieures comme l'analyse et la recherche d'images. Cependant le traitement des histogrammes comme vecteurs de fréquences ne tient pas compte de la localisation des cellules tandis que, pour certaines mesures de proximité entre deux distributions comme l'*Earth Mover's Distance* [RTG98] ou la distance de Mallows [Mal72], la localisation des *cellules* doit être prise en compte. Quand ces mesures sont utilisées, un histogramme est mathématiquement une collection de paires $\{(c_1, f_1), (c_2, f_2), \ldots, (c_k, f_k)\}$ (appelée aussi un *ensemble de vecteurs pondérés*), où $c_l \in \mathbb{R}^D$ est le centre de la cellule l. Un histogramme est ainsi une distribution particulière dans le sens où elle est discrète.

Lorsqu'un histogramme est considéré comme $\{(c_l, f_l)\}_{l=1,2,\ldots,k}$, une extension directe de l'histogramme est de générer de façon adaptative les c_l et f_l et de laisser le nombre de cellules, k, dépendre de l'image en question. Il s'agit des signatures basées sur les régions utilisées dans [DMK+01]. Soit $S_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,n_i}\}$ un ensemble de descripteurs

locaux d'une image i. L'application d'un algorithme de *clustering*, par exemple, k-means, sur l'ensemble S_i regroupe les descripteurs en k_i *clusters*. Si l'on note $c_{i,l}$ le centre de gravité du cluster l, la signature de l'images i sera $\{(c_{i,1}, f_{i,1}), \ldots, (c_{i,k_i}, f_{i,k_i})\}$. Le nombre de clusters, k_i, varie d'une image à l'autre.

Notons que les distributions extraites à partir d'un ensemble de descripteurs locaux peuvent être sous d'autres formes, par exemple, une densité de probabilité continue [DV02] ou un modèle stochastique spatial [LW04].

1.2.4 Discussion

Diverses méthodes pour l'extraction de caractéristiques visuelles et la construction de signatures d'images ont été proposées. Alors que les caractéristiques globales sont appropriées pour décrire le contenu global d'une image entière, les caractéristiques locales conviennent pour représenter les détails. Donc, le choix d'une représentation appropriée dépend de l'échelle du contenu principal. Dans ce sens, une représentation hybride pourrait être intéressante.

Il y a eu, au cours des années, un passage de la *représentation globale* comme les histogrammes de couleurs, les descripteurs de formes globaux vers la *représentation locale* avec des caractéristiques et descripteurs locaux comme les points d'intérêt, les caractéristiques basées sur les régions, les modèles spatiaux et la caractérisation de formes locales. Les caractéristiques locales correspondent souvent aux composantes les plus significatives de l'image. Cela permet l'association directe de la sémantique avec les composants (régions, objets) de l'image. Bien que la segmentation soit destinée à reconnaître des objets dans une image, la segmentation précise reste encore un problème ouvert. Par conséquent, d'autres approches alternatives comme les points d'intérêt se sont révélées plus efficaces. Lorsque les descripteurs locaux sont utilisés pour décrire le contenu de l'image, leurs distributions peuvent être explorées soit de façon discrète par exemple par *quantification vectorielle* ou *représentation basée sur les régions*, soit de façon continue avec des densités de probabilité ou de manière plus complexe avec des modèles stochastiques spatiaux. Tandis que la *quantification vectorielle* utilise un nombre fixe de cellules pour toutes les images, ne tient pas compte de leur localisation et les traite comme des variables symboliques (mots visuels), la *représentation basée sur les régions* génère les cellules de façon adaptative à chaque image. La localisation (représentant) des cellules est alors prise en compte pour mesurer la similarité entre des images.

Les informations spatiales sont soit intégrées dans la phase de construction de signatures pour former des représentations complexes, soit utilisées dans l'étape de post-traitement qui vérifie la cohérence de la mise en correspondance des descripteurs locaux.

En résumé, réduire le *fossé sensoriel* en tandem avec le *fossé sémantique* doit continuer à être un but de la description du contenu visuel dans le futur.

1.3 Similarité des images

Après avoir choisi les signatures d'image, comment les utiliser pour une recherche d'images similaires à l'image requête ? Il faut définir une *mesure de similarité*[2]. Malheureusement, il n'existe pas de définition générale de la mesure de similarité car elle dépend des besoins de chaque application. Cependant, n'importe quelle mesure de similarité prend deux objets comme paramètres d'entrée et détermine un nombre réel non-négatif qui traduit la similarité entre les deux objets. Une mesure de similarité est donc une fonction :

$$sim : Obj \times Obj \rightarrow \mathbb{R}^+$$

Le choix de mesures de similarité dépend des types de signatures utilisés et de la similarité recherchée. Les mesures de similarité utilisées dans la RIC sont discutées dans [SWS+00] et [DJLW08]. Avant de donner un bref aperçu des mesures les plus couramment utilisées, nous rappelons quelques définitions.

Définition 1.3 (Métrique) - *Une **métrique** sur un ensemble $\mathbb{E}$, est une fonction (appelée **fonction de distance** ou simplement **distance**)*

$$\begin{array}{rcl} d : \mathbb{E} \times \mathbb{E} & \rightarrow & \mathbb{R}^+ \\ (\mathbf{x}, \mathbf{y}) & \mapsto & d(\mathbf{x}, \mathbf{y}) \end{array}$$

(où $\mathbb{R}^+$ est l'ensemble des nombre réels non négatifs) vérifiant $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{E}$:

1. $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} \equiv \mathbf{y}$ *(identité)*
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ *(symétrie)*
3. $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$ *(inégalité triangulaire)*

*On dit alors que $(\mathbb{E}, d)$ est un **espace métrique**.*

Notons que la définition d'un espace métrique n'implique pas que les signatures s'expriment sous forme de vecteurs. Ainsi, la distance peut être une fonction quelconque comparant directement deux images.

La figure 1.4 résume les principaux types de signatures, les distances et les techniques associées. Dans la suite, pour chaque type de signatures, nous clarifions sa description mathématique qui conditionne le choix de distances, et l'emploi de telle ou telle technique. Nous commençons par le type de signature le plus simple, le vecteur de caractéristiques.

1.3.1 Signatures de type « vecteurs »

Lorsque les images sont représentées par de simples vecteurs de caractéristiques, la mesure de dissimilarité la plus utilisée est sans doute la *distance euclidienne* (ou norme

2. Nous parlerons de mesures de similarité dans le cas général et plus spécifiquement de mesure de dissimilarité dans le sens où la mesure est nulle pour deux objets identiques et croissante lorsque les objets sont de moins en moins similaires. À ce sens une distance est une mesure de dissimilarité.

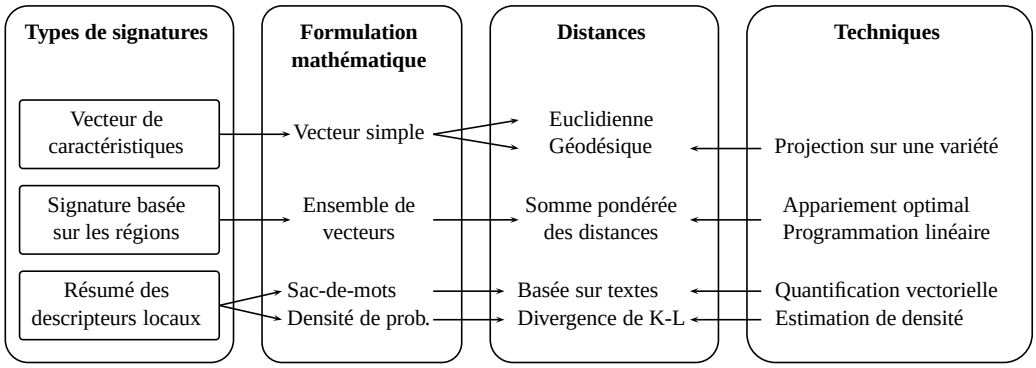


FIGURE 1.4 – Différents types de signature d'images, leur formulation mathématique, les distances utilisées, et les techniques reliées à la formulation de signatures et au calcul des distances.

L_2). Soient $\mathbf{x}, \mathbf{y} \in \mathbb{E} \equiv \mathbb{R}^D$ deux vecteurs dans l'espace de dimension D, la distance euclidienne entre $\mathbf{x}$ et $\mathbf{y}$ est définie de manière suitvante :

$$L_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^{D} (x_j - y_j)^2} \tag{1.1}$$

où x_j est la $j^{ième}$ composante de $\mathbf{x}$ et y_j est la $j^{ème}$ composante de $\mathbf{y}$.

La distance euclidienne est un cas particulier d'une distance plus générale, la *distance de Minkowski* (ou norme L_p). Soient $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$ deux vecteurs dans l'espace de dimension D, la distance de Minkowski d'ordre p entre $\mathbf{x}$ et $\mathbf{y}$ est obtenue par :

$$L_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^{D} |x_j - y_j|^p \right)^{\frac{1}{p}} \tag{1.2}$$

où $|.|$ désigne la valeur absolue. Dans le cas où $p = 2$, nous avons la distance euclidienne (L_2) et la *distance de Manhattan* (L_1) dans le cas où $p = 1$. Quand p tend vers l'infini nous obtenons la *distance de Tchébychev* :

$$L_\infty(\mathbf{x}, \mathbf{y}) = \lim_{p \to \infty} \left(\sum_{j=1}^{D} |x_j - y_j|^p \right)^{\frac{1}{p}} = \max_{j=1}^{D} |x_j - y_j| \tag{1.3}$$

Pour gagner en robustesse, de nombreuses alternatives ont été proposées aux distances de Minkowski (L_p). Une des plus connues est la *distance de Mahalanobis* [Mah36] qui permet de tenir compte des incertitudes sur les vecteurs ainsi que la corrélation éventuelle de leurs composantes. La distance de Mahalanobis de deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ avec une matrice de variance-covariance Σ est donnée par :

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^{\mathrm{T}} \, \Sigma^{-1} \, (\mathbf{x} - \mathbf{y})} \tag{1.4}$$

où $\mathbf{x}^{\mathrm{T}}$ est la transposée de $\mathbf{x}$.

La qualité des distances dans un espace supposé linéaire est souvent moins bonne que si on travaillait dans un espace non-linéaire. Une façon de résoudre ce problème consiste à rechercher une variété non-linéaire où se trouvent les vecteurs et à remplacer la distance euclidienne par la *distance géodésique*. On suppose ici que le sous-espace non-linéaire est plus adapté à la perception visuelle que l'espace linéaire original. La similarité peut donc être plus pertinente si elle est calculée dans la variété non-linéaire. Cette idée a été appliquée au *ranking* d'images [He04, HLZ+04, HMZ04, ZWG+04, VL05]. Les méthodes typiques pour l'apprentissage basé sur les variétés sont *locally linear embedding* (LLE), *isomapping*, et *multidimensional scaling* (MDS) [dST03].

1.3.2 Signatures de type « ensembles de vecteurs »

Les signatures basées sur les régions et les histogrammes (dans le cas où la localisation des *cellules* est prise en compte) peuvent être considérées comme des *ensembles de vecteurs*, plus précisément des *ensembles de vecteurs pondérés*. Les mesures de dissimilarité pour ce type de signatures utilisent la **distance entre deux ensembles de vecteurs** qui est moins évidente que la distance entre deux vecteurs simples.

D'abord, considérons une signature d'image sous forme d'un *ensemble de vecteurs pondérés*, $\{(c_1, f_1), (c_2, f_2), \ldots, (c_n, f_k)\}$ où c_l est le vecteur de caractéristiques de la régions l ou le représentant du cluster l ou encore la localisation de la *cellule* l (cf. Section 1.2.3) avec son poids (ou fréquence) associé, f_l. Soient deux signatures notées :

$$S^{(m)} = \left\{ \left(c_1^{(m)}, f_1^{(m)}\right), \left(c_2^{(m)}, f_2^{(m)}\right), \ldots, \left(c_{k_m}^{(m)}, f_{k_m}^{(m)}\right) \right\}, m = 1, 2.$$

Une approche naturelle pour définir la mesure de similarité entre deux signatures est d'apparier les $c_i^{(1)}$ et les $c_j^{(2)}$ et puis de combiner les distances entre ces vecteurs comme une distance entre deux ensembles de vecteurs. Une méthode pour l'appariement des vecteurs est d'associer un poids $a_{i,j}$ à chaque paire $(c_i^{(1)}, c_j^{(2)})$, $1 \leq i \leq k_1, 1 \leq j \leq k_2$ [WLW01]. Le poids $a_{i,j}$ indique l'importance de l'association du vecteur $c_i^{(1)}$ au vecteur $c_j^{(2)}$. Ces poids $a_{i,j}$ doivent vérifier certaines contraintes :

$$\begin{aligned} \sum_j a_{i,j} &= f_i^{(1)} \\ \sum_i a_{i,j} &= f_j^{(2)} \end{aligned}$$

Une des motivations pour cet *appariement flou* (*soft matching*) est de diminuer l'effet de la segmentation d'images incorrecte sur la qualité de la recherche d'images. Une fois les poids déterminés, la distance entre $S^{(1)}$ et $S^{(2)}$ est agrégée à partir des distances entre des paires de vecteurs :

$$d_{ens.}(S^{(1)}, S^{(2)}) = \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} a_{i,j}\, d\left(c_i^{(1)}, c_j^{(2)}\right), \tag{1.5}$$

où la distance $d(.,.)$ est une distance quelconque entre deux vecteurs simples.

Une heuristique pour déterminer les poids $a_{i,j}$ est de chercher les $a_{i,j}$ tels que la distance $d_{ens.}(S^{(1)}, S^{(1)})$ dans l'équation 1.5 soit minimisée sous certaines contraintes sur les $a_{i,j}$.

$$d_{ens.}(S^{(1)}, S^{(2)}) = \min_{a_{i,j}} \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} a_{i,j}\, d\left(c_i^{(1)}, c_j^{(2)}\right), \tag{1.6}$$

sous les contraintes :

$$\sum_{j=1}^{k_2} a_{i,j} = f_i^{(1)}, \forall i = 1, 2, \ldots, k_1, \tag{1.7}$$

$$\sum_{i=1}^{k_1} a_{i,j} = f_j^{(2)}, \forall j = 1, 2, \ldots, k_2, \tag{1.8}$$

$$a_{i,j} \geq 0, \forall i = 1, 2, \ldots, k_1; j = 1, 2, \ldots, k_2. \tag{1.9}$$

Cette distance est exactement la distance de Mallows dans le cas de distributions discrètes [Mal72].

Une autre méthode d'appariement est la *distance de Hausdorff* qui est utilisée dans [KB02] pour la recherche d'images. Chaque vecteur $c_i^{(1)}$ dans $S^{(1)}$ est apparié à son plus proche vecteur dans $S^{(2)}$, dit $c_{i*}^{(2)}$, et la distance entre $S^{(1)}$ et $S^{(2)}$ est le maximum de toutes les $d(c_i^1, c_{i*}^{(2)})$. La distance de Haussdorff est symétrisée par calculer une autre distance supplémentaire en changeant le rôle de $S^{(1)}$ et $S^{(2)}$ et choisir la distance la plus longue :

$$d_H(S^{(1)}, S^{(2)}) = \max\left(\max_i \min_j d\left(c_i^{(1)}, c_j^{(2)}\right), \max_j \min_i d\left(c_j^{(2)}, c_i^{(1)}\right)\right) \tag{1.10}$$

D'autres distances basées sur les techniques de mise en correspondance floue très utilisées pour les signatures de type d'ensemble de vecteurs sont l'EMD (Earth Mover's Distance) [RTG00] et l'IRM (Integrated Region Matching) [LWW00]. L'EMD consiste à minimiser le coût de transformation d'une distribution en une autre sous certaines contraintes de déplacement. Quand les $f_i^{(1)}$ et $f_j^{(2)}$ sont des probabilités (par ex. histogrammes normalisés, densités de probabilité), l'EMD est équivalente à la distance de Mallows. La distance IRM utilise le principe « plus similaire plus grande priorité » (ou MSHP pour *Most Similar Highest Priority* en anglais) pour l'appariement des régions. Le principe est que plus la distance entre une paire de régions $d\left(c_i^{(1)}, c_j^{(2)}\right)$ est courte plus le poids correspondant $a_{i,j}$ est grand.

1.3.3 Signatures de type « résumé des descripteurs locaux »

Les descripteurs locaux sont résumés soit de façon discrète par des histogrammes de mots visuels ou des signatures basées sur les régions, soit de façon continue par des densités de probabilité (cf. Section 1.2.3).

Lorsqu'une quantification vectorielle est utilisée pour créer un *vocabulaire visuel*, les images peuvent être représentées par un modèle vectoriel avec une pondération *tf*idf* [SB88] ou plus simplement par un modèle d'histogramme de mots visuels [ZZRS00, SZ03b, NS06, JHS07, LS07]. Avec le modèle vectoriel, la mesure de similarité fréquemment utilisée est la *similarité du cosinus* qui permet de mesurer la similarité entre deux vecteurs en déterminant le cosinus de leur angle. Soient $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$ deux vecteurs à D dimensions, la similarité entre $\mathbf{x}$ et $\mathbf{y}$ est obtenue par :

$$\begin{aligned} sim_{mv}(\mathbf{x}, \mathbf{y}) &= \cos(\mathbf{x}, \mathbf{y}) \\ &= \frac{\langle \mathbf{x} \cdot \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|} \end{aligned} \tag{1.11}$$

où $\langle \mathbf{x} \cdot \mathbf{y} \rangle$ est le produit scalaire de deux vecteurs $\mathbf{x}$ et $\mathbf{y}$; et $\|\cdot\|$ est la norme euclidienne. Pour le modèle d'histogramme, une mesure de similarité a été proposée dans [ZZRS00] :

$$sim_{mh}(\mathbf{x}, \mathbf{y}) = \frac{1}{1 + dist_{mh}(\mathbf{x}, \mathbf{y})} \tag{1.12}$$

où la distance est :

$$dist_{mh}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{D} \frac{|x_i - y_i|}{1 + x_i + y_i} \tag{1.13}$$

On peut aussi estimer les densités de probabilité des distributions de descripteurs locaux et les utiliser comme des signatures. La mesure de dissimilarité entre deux images est alors la différence entre deux distributions représentées par leurs densités de probabilités. Une mesure de différence très utilisée est la divergence de Kullback-Leibler (divergence K-L) [KL51]. La divergence K-L, connue comme l'*entropie relative*, est une mesure asymétrique de deux distributions $f(.)$ et $g(.)$, définie par :

$$d_{\mathrm{KL}}(f, g) = \int_{-\infty}^{+\infty} f(x) \log \frac{f(x)}{g(x)} dx \tag{1.14}$$

$$d_{\mathrm{KL}}(f, g) = \sum_{x} f(x) \log \frac{f(x)}{g(x)} \tag{1.15}$$

dans les cas continus et discrets respectivement [DV02, MSB02].

Discussion

En bref, le calcul de similarité peut être effectué avec des vecteurs de caractéristiques simples, des ensembles de vecteurs et des descripteurs locaux résumés. L'avantage principal de la représentation des images par de *simples vecteurs* est l'efficacité des opérations algébriques et géométriques sur ces vecteurs. Cependant, plusieurs de ces représentations ne capturent pas les détails nécessaires pour représenter la sémantique complexe des images. C'est pour cela que les signatures basées les descripteurs locaux ont été

proposées. D'un autre côté, la complexité de calcul des mesures de similarité sur les ensembles de descripteurs complique le problème de la recherche d'images. Le résumé des descripteurs locaux en histogrammes (avec les mesures de similarité associées) est un compromis entre l'efficacité (les détails d'images sont pris en compte) et la parcimonie (la représentation simple rend la recherche rapide) d'un système RIC.

1.4 Méthodes inspirées de l'analyse de données textuelles

Lorsqu'un modèle « sac-de-mot-visuels » est utilisé pour décrire des images, les méthodes développées pour l'analyse de données textuelles peuvent être appliquées aux images. Nous présentons dans cette section quelques méthodes inspirées de l'analyse de données textuelles : LSA (Latent Semantic Analysis), PLSA (Probabilistic Latent Semantic Analysis) et LDA (Latent Dirichlet Allocation). Ces méthodes ont été développées à l'origine pour traiter des données textuelles (les documents) où le modèle de « sac-de-mots » est utilisé. Un document est décrit par la fréquence des mots qui apparaissent dans le document. La représentation d'un corpus (l'ensemble des documents) forme un tableau de contingence qui croise les documents et les mots.

Soit $\mathbf{X}$ le tableau de contingence à M lignes et N colonnes obtenu en ventilant M documents d'un corpus $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ et N mots d'un vocabulaire $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$. L'élément x_{ij} du tableau $\mathbb{F}$ désigne le nombre de fois où le mot v_j apparaît dans le document d_i.

1.4.1 LSA

Le LSA [DDF+90] est une méthode qui projette les documents $d_i \in \mathcal{D}$ dans un espace réduit, appelé l'*espace latent*. Le LSA utilise une décomposition en valeurs singulières de la matrice de données (c-à-d. le tableau de contingence $\mathbf{X}$ avec ou sans pondération *tf*idf*) pour identifier un sous-espace linéaire qui capture la plupart des variances dans le corpus :

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{1.16}$$

où $\mathbf{U}$ et $\mathbf{V}$ sont des matrices orthogonales $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$ et $\mathbf{\Sigma}$ est une matrice diagonale dont les éléments diagonaux sont les valeurs singulières de $\mathbf{X}$.

L'approximation de $\mathbf{X}$ (par LSA) est calculé en mettant à zéro les $(N - K)$ plus petites valeurs singulières ($\tilde{\mathbf{\Sigma}}$).

$$\begin{aligned} \tilde{\mathbf{X}} &= \mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^{\mathrm{T}} \\ &\approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \\ &\approx \mathbf{X} \end{aligned} \tag{1.17}$$

La nouvelle représentation des documents dans l'espace latent est donnée par $\mathbf{U}\tilde{\mathbf{\Sigma}}$.

Cette approche peut servir à une compression significative de grandes collections. De plus, Deerwester *et al.* affirment que les axes du LSA, qui sont des combinaisons linéaires

de dimensions originales peuvent capturer certains aspects des notions linguistiques comme le *synonymie* et la *polysémie*.

1.4.2 PLSA et LDA

Une version probabiliste du LSA est le PLSA (Probabilistic Latent Semantic Analysis) [Hof99a, Hof99b] qui améliore le LSA en introduisant un modèle probabiliste : la distribution des mots dans un document est considérée comme multinomiale. La méthode se base sur une décomposition des mélanges dérivée d'un modèle de variables latentes en utilisant l'algorithme d'EM [DLR77] pour estimer les paramètres.

En effet, le PLSA introduit une variable latente $z \in \mathcal{Z} = \{z_1, z_2, ..., z_K\}$ et modélise la probabilité jointe $P(d, v)$, $d \in \mathcal{D}$ et $v \in \mathcal{V}$ par :

$$P(d,v) = P(d)P(v|d) \tag{1.18}$$

$$\text{avec } P(v|d) = \sum_{z\in\mathcal{Z}} P(v|z)P(z|d) \tag{1.19}$$

Le logarithme de la vraisemblance du corpus est défini par :

$$\mathcal{L} = \sum_{d\in\mathcal{D}}\sum_{v\in\mathcal{V}} \mathbf{X}(d,v)\log(P(d,v)) \tag{1.20}$$

où $\mathbf{X}(d, v)$ est le nombre d'occurrences du mot v dans le document d.

$\mathcal{L}$ est maximisé par un algorithme d'EM avec l'étape E :

$$P(z|d,v) = \frac{P(z)P(d|z)P(v|z)}{\sum_{z'} P(z')P(d|z')P(v|z')} \tag{1.21}$$

suivie par l'étape M :

$$P(d|z) = \frac{\sum_v \mathbf{X}(d,v)P(z|d,v)}{\sum_{d',v} \mathbf{X}(d',v)P(z|d',v)} \tag{1.22}$$

$$P(v|z) = \frac{\sum_d \mathbf{X}(d,v)P(z|d,v)}{\sum_{d,v'} \mathbf{X}(d,v')P(z|d,v')} \tag{1.23}$$

$$P(z) = \frac{1}{x_{..}}\sum_{d,v} \mathbf{X}(d,v)P(z|d,v), \text{ avec } x_{..} \equiv \sum_{d,v} \mathbf{X}(d,v). \tag{1.24}$$

Chaque valeur de la variable latente $z \in \mathcal{Z}$ correspond à un « *thème* ». Un mot est généré à partir d'un thème (avec la distribution des mots par thème $P(v|z)$) et des mots différents dans un document peuvent être générés de différents thèmes. Ainsi, un document peut appartenir à plusieurs thèmes. Le PLSA représente chaque document par sa distribution des thèmes $P(z|d)$.

Lorsqu'adapté aux images, [WAC+04, SRE+05, LS07], les intérêts du PLSA sont la réduction de dimension et l'amélioration de la représentation sémantique des images.

Bien que le PLSA soit une méthode utile pour la modélisation probabiliste des données textuelles, il ne fournit aucun modèle probabiliste au niveau des documents.

En effet, dans le PLSA, un document est représenté par une distribution des thèmes, $P(z|d)$. Il faut bien noter que d n'est autre qu'un index de la liste des documents dans l'ensemble d'apprentissage et que le PLSA estime les distributions $P(z|d)$ pour ces documents seulement. Ceci conduit à deux problèmes : (i) le nombre de paramètres du modèle augmente linéairement avec la taille du corpus, qui souffre parfois du *sur-apprentissage* ; et (ii) le PLSA ne permet pas de générer de nouveaux documents[3]. Pour cette raison il n'est pas un « vrai » modèle génératif.

Pour remédier à ce problème, Blei *et al.* a proposé le LDA (Latent Dirichlet Allocation) [BNJ03] qui utilise un vrai modèle génératif. Le nombre de paramètres du modèle est indépendant de la taille du corpus. Cependant, le modèle est si complexe que l'inférence exacte pour estimer ses paramètres n'est pas possible. Donc, des méthodes approximatives variationnelles ont été développées [BNJ03] pour estimer les paramètres du modèle.

Le PLSA est un cas particulier du LDA quand un estimateur *maximum a posteriori* est utilisé et que la distribution de Dirichlet est supposée uniforme [GK03].

Signalons que le nombre de thèmes pour le PLSA et le LDA ne peut pas être très grand en général. Sinon, les probabilités sont trop petites et ne sont pas significatives. En plus, l'algorithme d'EM ne donne pas toujours une solution globale mais plutôt une solution locale.

1.5 Interaction homme - machine

Pour la recherche par le contenu, l'interaction de l'utilisateur avec le système de recherche est cruciale puisque les requêtes et leurs modifications ne peuvent être obtenues que par la participation de l'utilisateur. Les interfaces utilisateur dans les systèmes de recherche d'images se composent typiquement d'une interface de formulation de la requête et d'une autre qui sert à la présentation des résultats.

1.5.1 Spécification de requête

Préciser quelles images un utilisateur souhaite rechercher dans une base d'images de données peut se faire de plusieurs façons. Les formulations de requêtes les plus utilisées sont : la navigation par catégorie (*category browsing*), la *requête par concept*, la *requête par crayonnage* (*sketch*), et la *requête par exemple*. La *navigation par catégorie* consiste à naviguer dans la base d'images selon les catégories d'images. Dans ce but, les images de la base sont classifiées en différentes catégories selon leur sémantique ou leur contenu visuel [VFJZ01]. La *requête par concept* vise à rechercher les images selon la description conceptuelle associée aux images de la base. Dans la *requête par crayonnage* [KKOH92] et la *requête par exemple* [ABP00], l'utilisateur spécifie la requête en dessinant une esquisse ou en fournissant une image exemple et les images avec des caractéristiques visuelles similaires à la requête sont alors proposées comme réponses. Les deux premiers

3. Hofmann a proposé dans [Hof99a] une méthode pour calculer les distribution des thèmes $P(z|d)$ pour des nouveaux documents en lançant encore un algorithme d'EM qui fixe les distributions des mots par thème $P(v|z)$ obtenues dans l'étape d'apprentissage.

types de requêtes sont reliés à la description sémantique d'images qui sont hors de notre discussion. Nous nous concentrons seulement sur les deux derniers types de requête.

La requête par crayonnage permet à l'utilisateur de dessiner une esquisse de l'image avec un éditeur graphique fournit par le système RIC ou par un autre logiciel. Les requêtes peuvent être formulées en dessinant plusieurs objets avec certaines propriétés comme couleur, texture, forme. Dans la plupart des cas, une esquisse grossière est suffisante lorsque la requête peut être raffinée à partir des résultats.

La requête par exemple permet à l'utilisateur de formuler une requête en fournissant une image exemple. Le système extrait d'abord les caractéristiques de l'image requête et la décrit par une représentation interne (cf. Section 1.2). Puis, les images de la base ayant des caractéristiques similaires à celles de la requête sont renvoyées. L'avantage principal de la requête par exemple est qu'il n'est pas nécessaire pour l'utilisateur de fournir une description explicite de la cible, cette description étant calculée par par le système. Ce type de requête convient aux applications où la cible est une image avec un même objet ou un ensemble d'objets sous différentes conditions de projection. La plupart des systèmes courants fournissent ce type de formulation de la requête.

La requête par groupe d'images exemples permet à l'utilisateur de choisir de multiples images. Le système trouvera les images qui répondent le mieux aux caractéristiques du groupe d'exemples. De cette manière, une cible peut être définie plus précisément en spécifiant les variations des caractéristiques pertinentes et en supprimant les variations non pertinentes dans la requête. En plus, les propriétés de groupe peuvent être raffinées en introduisant des exemples négatifs. Plusieurs systèmes développés récemment fournissent des réponses à des requêtes contenant des exemples positifs et négatifs.

1.5.2 Visualisation

La présentation des résultats de recherche d'images est peut-être un des facteurs importants dans l'acceptation et la popularité d'un système de recherche d'images. Nous caractérisons quelques schémas de visualisation courants :

- *Ordonné par pertinence.* La façon la plus populaire pour présenter les résultats de recherche est l'arrangement par pertinence, adopté par Google et Yahoo ! pour leurs moteurs de recherche d'images. Les résultats sont ordonnés selon une mesure numérique de pertinence à la requête.
- *Ordonné dans le temps* (*chronologique*). Les images retournées apparaissent dans un ordre chronologique. Le système Google's Picasa [4] pour des collections personnelles fournit une option pour visualiser ainsi les images dans de temps.
- *Groupé* (*clustering*). Le clustering d'images par leurs méta données ou leur contenu visuel a été un thème de recherche actif ces dernières années. Le clustering de résultats permet une forme de présentation intuitive et est aussi utilisé pour améliorer la performance du système [CWK05].
- *Hiérarchique.* Si les méta-données associées aux images peuvent être arrangées en arbre (par ex., WordNet [Mil95]), elles pourront fournir une aide utile dans la

4. `http://picasa.google.com`

visualisation. La visualisation hiérarchique des résultats est intéressante pour les archives, surtout dans des buts éducatifs.
- *Composé.* La combinaison se compose de deux ou plusieurs des types de visualisation mentionnés ci-dessus. Le clustering hiérarchique et la visualisation des graphes de concepts sont des exemples des techniques de visualisation composées.

1.5.3 Bouclage de pertinence

La perception humaine de la similarité entre images est subjective, sémantique, et dépendante des tâches. Bien que les méthodes basées sur le contenu fournissent des directions prometteuses pour la recherche d'images, les résultats basés uniquement sur la similarité des caractéristiques visuelles ne sont en générale ni perceptuellement ni sémantiquement satisfaisants. De plus, chaque type de caractéristiques visuelles a tendance à capturer un aspect seulement des propriétés des images et il est le plus souvent difficile pour l'utilisateur de spécifier clairement comment les différents aspects sont combinés. Pour résoudre à ces problèmes, le *bouclage de pertinence* (relevance feedback), une technique traditionnelle dans les systèmes de recherche d'information basés sur les textes, permet d'établir une faible liaison entre les concepts de haut niveau et les caractéristiques de bas niveau.

Le bouclage de pertinence est une technique d'apprentissage supervisé actif utilisée pour améliorer l'efficacité des systèmes de recherche d'information. L'idée principale est d'utiliser des exemples positifs et négatifs fournis par l'utilisateur pour améliorer la performance du système. Étant donnée une requête, le système retourne d'abord une liste d'images ordonnées selon une certaine mesure de similarité prédéfinie. Ensuite, l'utilisateur indique les images retournées comme pertinentes (exemples positifs) pour la requête ou non pertinentes (exemples négatifs). Le système raffine alors les résultats en basant sur le *feedback* et présentera une nouvelle liste d'images à l'utilisateur.

1.6 Métriques d'évaluation

La recherche d'images par le contenu est essentiellement un problème de recherche d'information. Les métriques d'évaluation adoptées naturellement sont celles qui sont utilisées en recherche d'information. Deux des mesures d'évaluation les plus populaires sont la *précision* et le *rappel* :
- *Précision* – Cette mesure se réfère au pourcentage des images retournées qui sont pertinentes par rapport à la requête.
- *Rappel* – Le *rappel* (ou la sensibilité) correspond au pourcentage de toutes les images pertinentes de la base d'images qui sont retournées.

Notons que quand la requête est une image, la pertinence des images retournées est extrêmement subjective. C'est pour cela, qu'au lieu de retourner un ensemble d'images pertinentes à la requête, la plupart des systèmes de recherche d'images retournent une liste d'images classées par pertinence décroissante par rapport à la requête. La *précision* et le *rappel* sont souvent calculés sur un certain ensemble de k premières images retournées. k est appelé le *scope*.

On a montré que la *précision* et le *rappel* suivent une relation inverse en fonction du *scope*, c'est-à-dire la précision diminue tandis que le rappel augmente quand le *scope* augmente.

Notons cependant qu'avec un *scope* k donné, la *précision* avec les k premières images retournées (dénotée par P@k) est proportionnelle au *rappel* (R@k) au même *scope*.

Traditionnellement, les résultats d'un système de recherche d'information sont résumés par des *courbes de précision–rappel* ou *courbes de précision–scope*.

Pour obtenir une *courbe de précision–rappel*, on calcule la *précision* à chaque image pertinente retournée et on interpole la *précision* à 11 points standards du *rappel*. Ce sont les points où le *rappel* est égal à 0, 0.1, 0.2, ...et 1. L'interpolation se fait par la règle suivante :

$$p(r) = \max_{r' \geq r} \{p(r')\}$$

où $p(r)$ est la *précision* au point où le *rappel* est égal à r.

Une *courbe de précision–rappel* idéale est parallèle à l'axe *rappel* et constant égale à 1 (c-à-d. la *précision* est toujours égale à 1 quelque soit le *rappel*).

Pour mesurer la manière dont le sytème ordonne des images pertinentes dans le résultat retourné à l'utilisateur, il y a deux mesures numériques (complémentaires des mesures graphiques comme les *courbes de précision–rappel* ou les *courbes de précision–scope*) très populaires dans la communauté de RIC. Ce sont la *précision moyenne* (*Average Precision*) et la mesure ANR (*Average Normalized Rank*).

La *précision moyenne* pour une requête est calculée comme l'aire sous la *courbe de précision–rappel* en moyennant les précisions à chaque image pertinente retournée. La moyenne arithmétique de la *précision moyenne* calculée sur un nombre de différentes requêtes est appelée le MAP (*Mean Average Precision*).

La mesure ANR pour une requête est donnée par :

$$ANR = \frac{1}{M * M_{per}} \left(\sum_{i=1}^{M_{per}} rang(i) - \frac{M_{per}(M_{per}+1)}{2} \right) \tag{1.25}$$

où M est le nombre d'images dans la base ; M_{per} est le nombre d'images pertinentes pour la requête et $rang(i)$ est le rang de la $i^{ème}$ image pertinente.

Essentiellement, l'ANR est égal à 0 si toutes les images pertinentes sont retournées les premières. La mesure ANR varie de 0 à 1, avec une valeur égale à 0.5 correspondant à une recherche aléatoire. Plus l'ANR est petite, meilleur est l'ordonnancement du résultat. Comme dans le cas de la *précision moyenne*, la moyenne arithmétique des ANR calculés pour plusieurs requêtes est appelée le $MANR$ (*Mean Average Normalized Rank*).

1.7 Synthèse

Nous avons présenté, dans ce chapitre, les principes de fonctionnement de la recherche d'images par le contenu, y compris la *description du contenu visuel*, les *mesures*

de similarités/dissimilarités (distances), les *méthodes inspirées de l'analyse de données textuelles*, l'*interaction homme-machine* et les *métriques d'évaluation*. Nous avons mis l'accent sur les techniques de description des caractéristiques visuelles et les mesures de similarités d'images basées sur les caractéristiques visuelles. Les détails de l'indexation des signatures d'images et les techniques de recherches efficaces sont présentés dans le chapitre 2.

Les caractéristiques visuelles générales les plus utilisées dans la recherche par le contenu sont : la couleur, la texture, la forme, et les relations spatiales. L'extraction des caractéristiques visuelles peut être réalisée soit globalement sur une image entière, soit localement sur un groupe de pixels. Ces deux approches de l'extraction des caractéristiques conduisent aux différents types de signatures d'images : vecteurs, ensembles de vecteurs, sac-de-mots-visuels, densités de probabilité de distributions des descripteurs locaux. Les relations spatiales entre des régions (ou objets) sont souvent représentées par un *2D-string* ou un modèle probabiliste. Les descripteurs locaux se sont révélés plus efficaces que les descripteurs globaux pour la description du contenu visuel.

Les techniques de calcul de mesures de similarités/distances entre les signatures d'images dépendent du type de signatures. Les distances entre des signatures de type « vecteur » sont faciles à calculer efficacement tandis que le calcul de distances entre des signatures de types « ensemble de vecteurs » est plus compliqué. Le résumé des descripteurs locaux en *sac-de-mots-visuels* est un bon compromis entre la *pertinence* (les descripteurs locaux sont utilisés pour la description des images) et la *rapidité* (le calcul des distances est très rapide) d'un système de recherche d'images par le contenu. Par ailleurs, les techniques de traitement des données textuelles peuvent être appliquées sur les images pour améliorer la qualité de la recherche. Ce sont aussi les raisons principales pour lesquelles nous avons choisi cette représentation d'images dans nos travaux de thèse.

Chapitre 2

Indexation et recherche d'images dans une base

2.1 Recherche d'images par leur signature

L'objectif d'un système de recherche d'images par le contenu est de retrouver les images dont les *signatures* sont les plus similaires à la signature requête au sens d'une certaine mesure de similarité (cf. Section 1.3). On parle alors de *recherche par similarité* qui est fondamentalement différente de la recherche exacte dans les bases de données classiques. Deux des principales méthodes de *recherche par similarité* sont les suivantes :

- *Recherche à un rayon près (ε-près).* La recherche à ε-près consiste à retrouve les images situées à une distance d'au plus ε de l'image requête.
- *Recherche des k-plus proches voisins.* Dans la recherche des k-plus proches voisins, il s'agit de retrouver les k images les plus proches de l'image requête selon une mesure de similarité associée aux signatures d'images.

Un des inconvénients de la recherche à ε-près est qu'on ne sait pas, a priori, combien d'images on va retrouver. Une valeur trop petite pour ε peut conduire à des résultats vides tandis qu'une valeur trop grande de ε peut donner des listes de résultats de très grande taille. En revanche, la recherche de k plus proches voisins garantit automatiquement k images dans la liste de résultats. Cependant, certaines images de la liste peuvent être très éloignées de la requête. Dans le cadre de la recherche d'images par le contenu, on préfère le plus souvent la recherche des k-plus proches voisins à la recherche à un rayon près bien que les deux approches possèdent chacune leurs avantages et leurs inconvénients. Lorsqu'une image est représentée par une *signature* unique, les k-plus proches voisins d'une signature correspondent directement aux k-plus proches images de l'image de requête. Cela permet à l'utilisateur de visualiser k images et d'évaluer la qualité de la recherche. Dans la suite, nous nous limitons à la recherche de k plus proches voisins.

L'approche naïve pour effectuer une recherche par similarité consiste à calculer, pour une signature de l'image requête, la mesure de similarité avec toutes les signatures de la base. On parlera alors d'une *recherche séquentielle* ou d'une *recherche exhaustive.*

La complexité de calcul d'une telle méthode est alors proportionnelle au nombre de signatures de la base, $O(M)$ où M est la nombre d'images de la base (ou la *taille de la base*). Lorsque la taille de la base est importante, une *recherche exhaustive* devient beaucoup trop coûteuse et on fait alors appel à des méthodes issues du domaines des bases de données pour accélérer les recherches. Le principe général est de limiter le nombre de signatures à comparer avec la signature requête en éliminant des paquets de signatures par des règles de filtrage. La performance des techniques d'indexation dépend beaucoup de la nature des signatures et des mesures de similarités utilisées. Nous présentons ci-dessous quelques techniques d'indexation pour les signatures de type vectoriel.

2.2 Indexation multi-dimensionnelle

Lorsque les signatures sont sous forme vectorielle, il est nécessaire de stocker les signatures dans une structure d'indexation de grande dimension et d'utiliser cette structure pour une recherche efficace.

2.2.1 Structure

Les méthodes d'indexation vectorielle sont basées sur le principe du regroupement hiérarchique de l'espace de données. Structurellement, elles ressemblent à un arbre de type B^+-tree [BM72, Com79]. Pour traiter les requêtes de manière efficace, les vecteurs sont stockés dans les feuilles d'un arbre, de telle sorte que les vecteurs qui sont proches l'un de l'autre dans l'*espace de signatures* sont présents dans une même feuille. Chaque vecteur est stocké dans une feuille unique, c-à-d. il n'y a pas de duplication de vecteurs. Les feuilles contenant des vecteurs sont organisées en arborescence structurée. Chaque nœud intermédiaire contient un ensemble de clés et de pointeurs vers un sous-arbre ou une feuille. Chaque clé correspond à un codage d'une région de l'espace multidimensionnel, dépendant de la structure utilisée. Dans un R-tree [Gut84], par exemple, la clé correspond à un hyper-rectangle englobant toutes les clés et tous les vecteurs du sous-arbre. Dans un SS-tree [WJ96], il s'agit d'une hyper-sphère.

Le fait de regrouper des vecteurs d'une même région de l'espace dans une même feuille permet de traiter la recherche en deux étapes. Une première étape consiste à déterminer les feuilles qui sont susceptibles de contenir des vecteurs répondant à la requête et la deuxième étape consiste à calculer la distance entre la requête et les vecteurs contenus dans les feuilles sélectionnées. Ceci permet de réduire sensiblement le nombre de vecteurs qui doivent être comparés à la requête.

Il y a un grand nombre de structures d'indexation développées pour la recherche efficace dans l'espace multidimensionnel. En général, les techniques d'indexation des données multidimensionnelles peuvent se classer en deux groupes : les *méthodes d'indexation basées sur le partitionnement des données* comme R-tree [Gut84], X-tree [BKK96], SS-tree [WJ96], SR-tree [KS97], et TV-tree [LJF94] divisent l'espace de données en fonction de la distribution de données ; et les *méthodes d'indexation basées sur le partitionnement de l'espace* comme grid-file [NHS84], K-D-B-tree [Rob81], et LSD^h-tree [Hen98] divisent

l'espace de données selon des hyperplans prédéfinis, sans tenir compte des valeurs actuelles des données, et stockent des vecteurs dans des cellules appropriées.

2.2.2 Algorithme de recherche

Avant de décrire en détail l'algorithme de recherche de k plus proches voisins utilisant les structures d'indexation multi-dimensionnelle, il est nécessaire de définir quelques notions importantes :

Définition 2.1 (MinDist et MaxDist) *Soit un vecteur de requête* $\mathbf{r}$ *dans un espace vectoriel* $\mathbb{E}$ *de dimension* D*. Soit* R *une région de* $\mathbb{E}$ *dépendant de la structure utilisée et caractérisée par sa clé dans l'arborescence. Soit* $d : \mathbb{E} \times \mathbb{E} \rightarrow \mathbb{R}^+$*, la distance considérée pour la recherche des* k *plus proches voisins de* $\mathbf{r}$*. Alors,*

$$\begin{aligned} MinDist(\mathbf{r}, R) &= \min_{\mathbf{x} \in R} d(\mathbf{x}, \mathbf{r}) \\ MaxDist(\mathbf{r}, R) &= \max_{\mathbf{x} \in R} d(\mathbf{x}, \mathbf{r}) \end{aligned}$$

Intuitivement, $MinDist$ est la plus petite distance entre une requête et une région et $MaxDist$ est la plus grande distance entre une requête et une région.

Les algorithmes de recherche des k plus proches voisins sont basés sur des règles de filtrage permettant d'éliminer directement des branches de l'arbre, lorsqu'il est impossible que la région correspondante contienne des points plus proches que la solution actuelle.

Il y a deux règles de filtrage. La première stipule que pour deux régions R_1 et R_2, si

$$MaxDist(\mathbf{r}, R_1) \leq MinDist(\mathbf{r}, R_2) \tag{2.1}$$

alors R_2 peut être éliminée. Cette règle suppose qu'il y ait au moins k vecteurs dans chaque région.

La seconde règle de filtrage stipule que si

$$MinDist(\mathbf{r}, R) \geq d(\mathbf{r}, \mathbf{x}_k) \tag{2.2}$$

où $\mathbf{x}_k$ représente le $k^{ème}$ plus proche voisin actuellement trouvé, alors R peut être éliminée.

Les deux algorithmes de recherche de k plus proches voisins les plus connus sont l'algorithme RKV [RKV95] et l'algorithme HS [HS95]. L'algorithme RKV est proposé originellement dans le cas du R-tree. C'est un algorithme de type « *branch and bound* » qui parcourt l'arbre en profondeur. La première règle de filtrage est renforcée par l'utilisation d'une distance, appelée $MinMaxDist$, plus restrictive que $MaxDist$. Elle correspond au minimum, sur les différents côtés de l'hyper-rectangle, de la $MaxDist$ de chaque côté. La règle de filtrage suppose alors que chaque côté de l'hyper-rectangle contient au moins un vecteur, ce qui est vrai dans le cas du R-tree par construction.

L'algorithme HS propose d'éviter de parcourir l'arbre de manière classique, en profondeur en en largeur d'abord. Pour cela il gère une liste de régions actives (LRA),

indépendamment du niveau de l'arbre auquel elles se situent. Cette liste est initialisée par la région correspondant à la racine de l'arbre (ainsi que sa MinDist), puis les opérations suivantes sont réitérées :

- Sélectionner la région R ayant la plus petite distance MinDist dans la LRA.
- Si R correspond à une feuille, alors chercher les plus proches voisins dans R.
- Sinon, calculer la MinDist des régions correspondant aux sous-arbres du nœud correspondant à R et les insérer dans la LRA.
- Supprimer R de la LRA.

L'avantage de l'algorithme HS est qu'il ne nécessite que la connaissance de MinDist. De plus, on a montré qu'il est plus performant en termes de nombre de nœuds ou de feuilles visités. L'inconvénient est qu'il nécessite plus de mémoire pour maintenir la LRA et que l'espace mémoire nécessaire dans le cas le pire est $O(n)$ où n est le nombre de régions.

2.2.3 Partitionnement des données

Toutes les méthodes d'indexation basées sur le partitionnement des données sont dérivées de la méthode R-tree [Gut84], développée originellement pour l'indexation des données $2D$ utilisées dans des systèmes d'information géographique. Le R-tree utilise les rectangles multidimensionnels (hyper-rectangles) englobants minimum (REM). Un REM est un intervalle multidimensionnel de l'espace de données (un rectangle multidimensionnel de côtés parallèles aux axes). Les REMs sont des approximations minimales des ensembles de points englobés. Donc, chaque surface de $(D-1)$ dimensions du REM contient au moins un vecteur [1]. Diverses méthodes peuvent être utilisées pour déterminer quels rectangles doivent être fusionnés ou séparés à chaque niveau de l'arbre. Le principal problème du R-tree est que les rectangles englobants peuvent se chevaucher. Par conséquent, la recherche doit parcourir plusieurs nœuds pour retrouver les plus proches voisins et devient inefficace. Le R^+-tree [SSH86, SRF87], une variante de R-tree, évite ce problème en utilisant une stratégie, appelée *forced-split*, qui sépare les régions de chevauchement en deux. Le nombre de nœuds peut augmenter de manière exponentielle. Le X-tree [BKK96], est une extension du R-tree développée pour gérer directement des données multidimensionnelles. Le chevauchement des régions est évité en utilisant un *split-history* et des *supernodes* avec des capacités élargies.

Le SS-tree [WJ96], une autre extension du R-tree, utilise des sphères englobantes au lieu de rectangles. Les sphères englobantes ne sont pas minimales, mais ce sont des sphères dont le centre est déterminé par le centre de gravité des points et dont le rayon est choisi tel que tous les vecteurs soient inclus dans la sphère. La description des régions comprend simplement un centre et un rayon. Ceci permet de déterminer rapidement les $MinDist$ et $MaxDist$.

Bien que les sphères améliorent la performance de la recherche, les sphères englobantes occupent plus d'espace que les rectangles englobants avec des données de grande dimension. Ceci réduit l'efficacité de la recherche. Par ailleurs, un problème général du

1. un point dans l'espace vectoriel

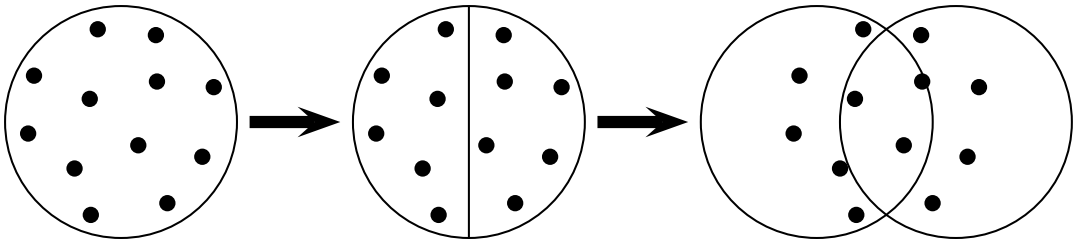


FIGURE 2.1 – Problème général du SS-tree : il n'y a pas de partitionnement sans chevauchement possible.

SS-tree est qu'il n'est pas toujours possible de faire un partitionnement sans chevauchement (cf. Figure 2.1).

Pour remédier à ce problème, le SR-tree [KS97] ne conserve que l'intersection d'une sphère englobante et d'un rectangle englobant. La raison d'une telle procédure, d'après les auteurs, est que les sphères sont plus appropriées que les rectangles pour rechercher les plus proches voisins en utilisant la métrique L_2. D'un autre côté, les sphères sont difficiles à maintenir et ont tendance à produire plus de régions de chevauchement. La combinaison sphère-rectangle permet de surmonter les deux inconvénients. La $MinDist$ (respectivement, $MaxDist$) est définie comme le maximum (respectivement, minimum) des $MinDist$ ($MaxDist$) du REM et de la sphère englobante. La $MinMaxDist$ n'est en pas définie pour les sphères, la $MinMaxDist$ des REMs est plutôt utilisée. Notons que les $MinDist$ et $MaxDist$ ne sont pas les $MinDist$ et $MaxDist$ exactes (c-à-d. la $MinDist$ exacte est inférieure à la $MinDist$ et la $MaxDist$ est supérieure à la $MaxDist$ exacte) et qu'aucune connaissance sur la sphère n'est exploitée pour déterminer la $MinMaxDist$.

Le TV-tree [LJF94] est une autre extension du R-tree. On utilise le fait que les dimensions ne jouent pas toutes le même rôle au cours de la recherche : certaines sont plus discriminantes que les autres. Ainsi, Lin *et al.* [LJF94] divisent les dimensions en trois classes : les dimensions que la recherche peut ignorer, les dimensions que la recherche doit utiliser, et les dimensions utilisées pour raffiner la recherche. L'inconvénient principal de cette technique est qu'elle requiert une connaissance exacte de la distribution de données selon chaque dimension et que l'on puisse utiliser certaines dimensions indépendamment des autres.

2.2.4 Partitionnement de l'espace

Les techniques de partitionnement de l'espace comme le grid-file [NHS84], le K-D-B-tree [Rob81] et le LSD$^{\text{h}}$-tree [Hen98] divisent l'espace en régions rectangulaires (cellules) disjointes plus ou moins régulières sans tenir en compte de la distribution de données. Les régions générées sont organisées soit sous forme d'une table de hachage comme dans le cas du grid-file, soit par une arborescence comme c'est le cas du k-d-tree [Ben75]. Contrairement aux méthodes de partitionnement de données, les méthodes de

partitionnement de l'espace garantissent un partitionnement disjoint et complet.

Cependant, un partitionnement complet génère en général des régions plus grandes que nécessaire. En particulier, dans un espace de grande dimension, le nombre de régions est beaucoup plus important que le nombre de vecteurs. Par conséquent, la plupart des régions sont vides. Le second problème est que les kd-tree ne sont généralement pas équilibrés. Le K-D-B-tree [Rob81] remédie à ce problème par une stratégie de « *forced-split* » comme dans le cas du R^+-tree.

Le LSD^h-tree [Hen98] est un kd-tree adaptatif. Contrairement aux R-trees et kd-trees, la description des régions dans les LSD^h-trees est encodée de manière sophistiquée pour réduire la mémoire nécessaire. Certains nœuds intermédiaires de haut niveau sont supposés être fixés dans la mémoire centrale pour accélérer la recherche. Un inconvénient majeur des méthodes de partitionnement de l'espace est que le nombre de régions voisines d'une région augmente exponentiellement avec le nombre de dimensions. De plus, la plupart de régions sont vides. Donc, la recherche devient inefficace quand le nombre de dimensions est grand ($D > 16$).

2.2.5 Courbes remplissant l'espace

Les courbes remplissant l'espace [Sag94] (*space filling curves* en anglais) appartiennent à la famille des méthodes de partitionnement de l'espace. Ces courbes comme les courbes de Hilbert, l'ordonnancement en Z ou les codes de Gray sont des applications d'un espace à D dimensions dans un espace à une dimension :

$$courbe : \mathbb{R}^D \rightarrow \mathbb{R}$$

L'utilisation d'approximations discrètes de ces courbes permet de réaliser un partitionnement de l'espace en cellules. À chaque étape, une dimension est choisie et l'espace est divisé en deux. La choix de la dimension dépend de la courbe utilisée. Si p est le nombre d'étapes que nous appelons la *profondeur du partitionnement*, prédéfini par l'utilisateur, l'espace sera divisé en 2^p cellules d'hypervolume égale et numérotées de 1 à 2^p. $cellule(\mathbf{x})$ est une fonction qui calcule la cellule dans laquelle le vecteur $\mathbf{x}$ se trouve.

$$cellule : \mathbb{R}^D \rightarrow [1; 2^p]$$

Les vecteurs sont transformés en entiers $[1; 2^p]$. N'importe quelle technique d'indexation uni-dimensionnelle est alors capable d'indexer les vecteurs. Supposons que le B^+-tree [BM72, Com79] soit utilisé.

Pour traiter les requêtes (k plus proches voisins ou à un rayon près), on considère les distances $MinDist$ et $MaxDist$ entre la requête et les cellules. Le problème se posant ici est comment évaluer ces distances de manière efficace sans énumérer toutes les cellules dans un intervalle donné. Il est possible de résoudre ce problème par un algorithme de type « *divide and conquer* » qui divise récursivement l'intervalle en deux. Bien que cet algorithme élimine un nombre significatif de cellules inutiles, le nombre de cellules à visiter augmente, en général, exponentiellement en fonction de p ($p \geq D$). Joly *et al.* [JBF05] ont étendu la recherche à ε-près en une *recherche statistique* dans laquelle les

cellules ayant une faible probabilité de contenir les réponses sont éliminées. L'efficience de la recherche s'améliore par rapport à la recherche à ε-près. Cependant, cette approche se base sur l'hypothèse que la *distribution des images distordues* engendrées par un *ensemble fini de transformations* est gaussienne avec indépendance entre les composantes. Ceci est supposé être vrai dans le cas de détection de copies. Par contre, dans le cas de recherche d'images, la distribution des images distordues n'est pas connue.

2.2.6 Pyramid-tree

Le pyramid-tree [BBK98] est une technique de partitionnement de l'espace conçue pour contrer la *malédiction de la dimension* [Bel61].

Deux phénomènes principaux permettent d'expliquer la dégradation des performances dans les espaces de grande dimension :

- *Équidistance.* Lorsque la dimension augmente, tous les vecteurs ont tendance à être équidistants les uns des autres. Le plus proche voisin d'une requête est aussi loin que tous les autres vecteurs dans la base. Il est nécessaire de comparer tous les vecteurs de la base avec la requête. Ainsi, une recherche séquentielle est plus appropriée.
- *Espace vide.* Plus la dimension augmente, plus l'espace semble vide. Le nombre de vecteurs contenus dans une sphère de rayon constant tend vers zéro rapidement lorsque la dimension augmente. Autrement dit, les vecteurs sont de plus en plus loin les uns des les autres.

Similaire aux courbes remplissant l'espace, le pyramid-tree transforme les vecteurs dans l'espace de D dimensions en clés dans l'espace à une dimension et utilise un B^+-tree pour indexer les clés. Contrairement aux autres méthodes, le pyramid-tree stocke les vecteurs et leur clé dans les feuilles du B^+-tree. Ainsi, aucune transformation inverse n'est nécessaire et l'étape de raffinement peut être effectuée sans demander d'autres fichiers. La partition dans un pyramid-tree consiste à diviser l'espace vectoriel en $2D$ pyramides dont le sommet est au centre de l'espace. Chaque pyramide est ensuite partitionnée grâce à des hyperplans parallèles à la base de la pyramide. De ce fait, le nombre de régions générées par la partition croît linéairement avec la dimension. Il ne souffre donc plus de la *malédiction de dimension.* La stratégie pour calculer les clés des vecteurs est optimisée pour la recherche à ε-près dans l'espace de grande dimension. Une requête à ε-près est traitée en déterminant toutes les pyramides ayant une intersection avec la requête et puis toutes les régions concernées, par un algorithme assez lourd comportant de nombreux tests des cas. On parcourt alors le B^+-tree et tous les vecteurs contenus dans les feuilles retrouvées sont comparés à la requête.

Le pyramid-tree est une structure d'indexation unique qui n'est pas affectée par la malédiction de la dimension. On a montré que, pour une distribution uniforme et une requête à un rayon près constant, les performances du pyramid-tree s'améliorent lorsque la dimension augmente [BBK98]. L'inconvénient majeur est qu'il ne permet de faire que des recherches à ε-près et non des recherches de k plus proches voisins.

2.3 Recherche séquentielle accélérée

Weber *et al.* ont montré dans [WSB98] que, selon certaines hypothèses pour une recherche de k plus proches voisins, la plupart des structures d'indexation multidimensionnelle devenaient moins performantes qu'une recherche séquentielle, lorsque le nombre de dimensions devenait supérieur à 16 environ.

2.3.1 VA-file

Suite à la constatation ci-dessus, Weber *et al.* ont proposé une technique de recherche par similarité, appelée VA-file [WSB98]. L'idée du VA-file est d'utiliser deux fichiers : un *fichier d'approximations* qui contient une version quantifiée, compressée des vecteurs et un *fichier original* qui contient les vrais vecteurs. Aucun des deux fichiers n'est trié. La quantification des vecteurs se fait en construisant une grille irrégulière sur l'espace des données. La résolution de la grille dans la dimension i correspond à 2^{b_i}, où b_i est le nombre de bits utilisés pour quantifier la dimension i. Le nombre de bits total pour quantifier un vecteur est alors de $b = \sum_{i=1}^{D} b_i$. Autrement dit, cette méthode partitionne l'espace en 2^b cellules.

La recherche de k plus proches voisins se fait en deux étapes. D'abord, les vecteurs quantifiés (se trouvant dans le premier fichier) sont chargés dans la mémoire et parcourus de manière séquentielle (étape de filtrage) et on applique les règles de filtrage classiques sur la $MinDist$ et la $MaxDist$ des cellules correspondantes. Les vecteurs candidats conservés sont ensuite ordonnés par leur $MinDist$ et leurs coordonnées sont chargées en mémoire par un accès direct au second fichier qui contient les vecteurs originaux (étape de raffinement). Cette étape s'arrête quand une $MinDist$ trouvée est supérieure ou égale à la distance entre la requête et le $k^{ème}$ plus proche voisin.

Fondamentalement, le gain du VA-File comparé à la recherche séquentielle est simplement le taux de compression, car la complexité du chargement séquentiel d'un gros fichier est linéaire en fonction de la longueur du fichier. La méthode est encore améliorée si le *fichier d'approximation* est contenu entièrement en mémoire principale. Cependant, le problème le plus important de cette méthode est l'étape de raffinement qui demande des accès aléatoires au disque. Avec une faible résolution dans la phase d'approximation (c-à-d. b est trop petit), le nombre de vecteurs candidats restants après l'étape de filtrage devient grand. Par conséquent, la performance gagnée par la compression est perdue. En revanche, si le nombre de bits utilisés pour la quantification est grand, le *fichier d'approximation* deviendra gros et il ne pourra pas être entièrement contenu en mémoire. Dans ce cas, le fichier doit être rechargé en mémoire à chaque requête et le coût engendré par le chargement du fichier et les calculs de la $MinDist$ et de la $MaxDist$ ne justifient pas son utilisation. Il est montré que, dans ce cas, le VA-file est à nouveau moins performant qu'à une recherche séquentielle [AG01].

La performance du VA-file dépend fortement de la qualité d'approximation car l'étape de filtrage se base sur l'approximation. Une amélioration du VA-file pour des distributions non-uniformes, appelée VA$^+$-file [FTAEA00], transforme des données dans le domaine de Karhunen-Loève (domaine de KL) par une analyse en composantes prin-

cipales (ACP). La quantification est effectuée dans le domaine de KL. Les pas de quantification sont déterminés par un algorithme *k-means*.

2.3.2 IQ-tree, GC-tree, et PRC-tree

Un inconvénient majeur du VA-file et du VA$^+$-file est que tous les vecteurs approximatifs doivent être examinés dans l'étape de filtrage. Pour réduire le nombre de vecteurs approximatifs à examiner pendant la recherche, quelques techniques qui combinent le VA-file et l'indexation basée sur un arbre ont été proposées.

Le IQ-tree [BBJ$^+$00] est un arbre d'indexation à trois niveaux organisé en trois fichiers distincts. Le premier niveau est une structure d'indexation arborescente (cf. Section 2.2.1) dont les feuilles contiennent des REMs et des pointeurs sur les pages du deuxième niveau qui contiennent des vecteurs quantifiés. La quantification se base sur la densité de vecteurs dans les REMs. Le troisième niveau du IQ-tree contient les pages de vecteurs originaux. Le GC-tree [CC02] partitionne l'espace de D dimensions en 2^D cellules pour identifier les *clusters* (régions denses) et les *outliers* (régions creuses) basés sur la densité locale des sous espaces. Les *clusters* sont récursivement partitionnés. Dans le GC-tree, l'approximation est utilisée pour représenter les régions denses tandis que pour une région creuse, les vecteurs originaux sont utilisés. Le PRC-tree [CZZ07] transforme les vecteurs dans le domaine de KL comme le VA$^+$-file et utilise ensuite un R-tree pour structurer les vecteurs quantifiés. La quantification utilise les premiers axes d'une ACP.

La recherche des k plus proches voisins dans le IQ-tree et le PRC-tree se fait en une seule étape. Les REMs sont filtrés en utilisant la $MinDist$. On peut accéder aux vecteurs originaux pendant la recherche. La différence entre IQ-tree et PRC-tree est que les REMs du IQ-tree sont définis sur l'espace original alors que les REMS du PRC sont calculés sur les vecteurs quantifiés. La recherche dans le GC-tree se fait en étapes comme dans le VA-file.

2.3.3 Fichier inversé

Lorsqu'une image est représentée par un vecteur creux, l'approche basée sur les fichiers inversés [SB88] est l'une des plus appropriées. Cette approche est initialement utilisée dans le contexte de recherche des textes où la requête est un ensemble de *mots clés*. Dès que le modèle de *sac-de-mots-visuels* est utilisé pour représenter les images, cette approche est aussi utilisée dans la recherche d'images [SZ03b, JHS07].

Supposons que tous les vecteurs représentant les images dans la base soient normalisés. La similarité cosinus entre une requête $\mathbf{r} = [r_1\ r_2\ \ldots\ r_D]$ ($\mathbf{r}$ est aussi normé) et une image $\mathbf{z} = [z_1\ z_2\ \ldots\ z_D]$ de la base est calculée par :

$$\begin{aligned} sim(\mathbf{r}, \mathbf{z}) &= \frac{\sum_{j=1}^{D} r_j\, z_j}{\|\mathbf{r}\|.\|\mathbf{z}\|} \\ &= \sum_{i=1}^{D} r_j\, z_j \end{aligned} \tag{2.3}$$

La similarité cosinus dans la formule 2.3 ne dépend que des $r_j \neq 0$. Pour une requête, seules les dimensions sur laquelle la coordonnée de la requête est non nulle sont utilisées. De plus, pour une dimension choisie, seules les images dont la coordonnée sur cette dimension est non nulle sont intéressantes. À partir de cette observation, un fichier inversé est construit pour chaque dimension j et contient des paires de $<i, z_{ij}>$ où i est la $i^{ème}$ image dans la base, et z_{ij} ($z_{ij} \neq 0$) est la coordonnée de l'image i sur la dimension j.

Le calcul des similarités pour une requête $\mathbf{r} = [r_1\ r_2\ \ldots\ r_D]$ et les images de la base consiste à prendre les fichiers inversés correspondants aux dimensions où $r_j \neq 0$. Pour chaque fichier inversé j (correspondant à la dimension j), la similarité entre la requête $\mathbf{r}$ et l'image i dans ce fichier est accumulée par une quantité de $(r_j z_{ij})$.

Cette technique est aussi généralisée dans [NS06] pour les distances de Minkowski d'ordre p quelconque en supposant que les vecteurs sont normés d'après la norme p. Rappelons la formule 1.2 pour calculer la distance de Minkowski d'ordre p dans le chapitre précédent.

$$\begin{aligned}
(L_p(\mathbf{z},\mathbf{r}))^p &= \sum_{j=1}^{D} |z_j - r_j|^p \\
&= \sum_{j|r_j=0} |z_j|^p + \sum_{j|z_j=0} |r_j|^p + \sum_{j|r_j\neq 0, z_j\neq 0} |z_j - r_j|^p \\
&= \|\mathbf{z}\|_p^p + \|\mathbf{r}\|_p^p + \sum_{j|r_j\neq 0, z_j\neq 0} (|z_j - r_j|^p - |r_j|^p - |z_j|^p) \\
&= 2 + \sum_{j|r_j\neq 0, z_j\neq 0} (|z_j - r_j|^p - |r_j|^p - |z_j|^p) \qquad (2.4)
\end{aligned}$$

La formule ne dépend aussi que des $r_j \neq 0$.

2.4 Recherche approximative

La dégradation des performances de la recherche exacte de plus proches voisins est liée à la grande dimension des vecteurs. Un moyen d'améliorer le temps de recherche consiste à recourir aux méthodes approximatives pour accélérer la recherche au détriment de la qualité des résultats. Notons cependant qu'une recherche exacte dans la base des signatures est donc déjà une approximation de la requête de l'utilisateur. Il est même des cas où l'introduction d'une erreur dans la qualité de recherche n'a aucun impact sur le résultat final de l'application. On peut séparer les méthodes de recherche approximatives en deux grandes catégories : celles où l'approximation porte sur la représentation des données et celles où elle porte sur la technologie de recherche.

2.4.1 Approximation de la représentation des données

L'approximation de la représentation des données consiste à transformer les données dans un espace de dimension plus petite que celle de l'espace original et à effectuer la recherche dans l'espace transformé. La technique de transformation la plus souvent utilisée est l'analyse en composantes principales [Pea01]. Une autre méthode est le hachage multidimensionnel [IM98, GIM99] qui offre un moyen de faire une recherche sur des points obtenus par une projection des vecteurs de l'espace original. L'idée consiste à transformer les vecteurs en utilisant plusieurs fonctions de hachage dans le but de maximiser la probabilité que des vecteurs proches de l'espace original se retrouvent dans un même paquet. Plusieurs transformations sont appliquées aux vecteurs et chacune d'elles associe la même valeur aux vecteurs proches. La mise en œuvre du hachage multidimensionnel proposée dans [GIM99] procède d'abord par une transformation des vecteurs de la base vers un cube de Hamming. La distance de Hamming est ensuite utilisée comme mesure de similarité et correspond à la distance L_1 dans l'espace original. De ce fait, l'approche ne peut s'appliquer qu'aux images dont la similarité est estimée par la distance L_1.

2.4.2 Approximation de la technique de recherche

Dans ce cas, nous distinguons globalement deux approches pour approcher la recherche de k-plus proches voisins pour réduire le temps de réponse. La première consiste à ramener le problème de la recherche des plus proches voisins à celui de la recherche à ε-près dans les structures d'indexation adaptées. C'est l'approche proposée dans [LS02]. Cette méthode estime d'abord un rayon ε pour k plus proches voisins sur un échantillon des données contenu en mémoire principale. Ce rayon est ensuite corrigé par un facteur pour minimiser la dépendance entre ε et l'échantillon choisi, et on obtient alors $\varepsilon_{corrigé}$. Une recherche à ε-près avec le rayon corrigé est effectuée sur toutes les données.

Soit R l'ensemble des vecteurs contenus dans les régions intersectées par l'hypersphère de rayon $\varepsilon_{corrigé}$. La dernière étape consiste donc à rechercher les k plus proches voisins dans R et les retourne comme une approximation de la recherche de k plus proches voisins. Cette technique réduit les coûts entrées/sorties pendant la recherche mais la qualité de la recherche dépend de l'échantillon choisi au départ. En effet, d'après les expérimentations présentées dans [LS02], plus la taille de l'échantillon est petite, plus l'erreur commise dans l'estimation de ε est grande. Cette erreur croît également avec le nombre de plus proches voisins. En plus, la taille de R n'est pas contrôlée, Il peut donc arriver qu'elle soit inférieure au nombre de plus proches voisins à calculer.

La seconde approche pour approcher la recherche des k plus proches voisins, beaucoup plus directement que la première, consiste à exploiter les propriétés des structures d'indexation dans le processus de recherche pour réduire le nombre de vecteurs à comparer avec la requête. Il s'agit de l'approximation géométrique des requêtes qui utilise les techniques de recherche exacte mais en modifiant la requête. Pour approcher une recherche des plus proches voisins, on a proposé la recherche de ε-plus proches voisins. Un ε-plus proche voisin est un vecteur dont la distance avec la requête est inférieure à

$(1+\varepsilon)$ fois la distance du plus proche voisin exact :

Définition 2.2 (ε-plus proche voisin) – *Soit $\Delta = \{S_i\}_{i\in[1;M]}$ une base de M signatures et $d : \Delta \times \Delta \to \mathbb{R}^+$ une mesure de dissimilarité. Une signature $S_{\varepsilon pp}$ est une ε-plus proche voisine d'une signature de requête S_r si :*

$$d(S_r, S_{\varepsilon pp}) \leq (1+\varepsilon) d(S_r, S_{pp})$$

où S_{pp} est la plus proche voisine exacte de S_r.

Cette requête est très facilement transposable à la plupart des méthodes exactes vues précédemment, puisqu'il suffit de modifier, dans les règles de filtrage (cf. section 2.2.2), la $MinDist$ par $MinDist + \varepsilon$ et la $MaxDist$ par $MaxDist - \varepsilon$. Dans le cas des structures d'indexation multidimensionnelle, cela revient à modifier la taille des régions englobantes, en diminuant ainsi le taux de chevauchement et en renforçant la sélectivité de l'étape de filtrage.

Les méthodes d'approximation géométrique reposent sur un principe très simple mais le choix de ε n'est pas facile pour faire un bon compromis entre le temps et la qualité de la recherche. Un petit ε préserve la précision mais ne garantit pas toujours une forte réduction du temps de réponse. En revanche, un grand ε réduit significativement le temps de recherche mais n'assure pas une bonne précision.

2.4.3 Clustering et classification

La classification et la catégorisation d'images sont souvent des étapes de prétraitement qui permettent de réduire le temps de réponse, d'améliorer la qualité des résultats de la recherche dans de grandes bases d'images ou d'effectuer une annotation automatique. Ce sont des approches d'approximation de la techniques de recherche. Le principe de ces approches est de regrouper les données de la base, lors d'une phase hors-ligne, via un algorithme de regroupement de données. Les agrégats ainsi formés, sont ensuite stockés individuellement sur le disque, dans un fichier. Lors de la recherche, une première étape sélectionne les agrégats les plus pertinents pour la requête, charge les fichiers correspondant en mémoire et traite séquentiellement les individus dans ces fichiers. L'approximation se fait au moment de la sélection des agrégats. Plus le nombre d'agrégats est grand, meilleure est la précision des résultats.

L'inconvénient principal de ce type d'approche est qu'elle n'est performante que lorsque les données sont effectivement fortement regroupées sous forme d'agrégats, ce qui n'est pas toujours le cas, y compris dans certaines applications de recherche d'images par le contenu. De plus, le coût de la phase hors-ligne de regroupement des données est assez élevé : ($O(M^2)$ en général) et peut rapidement devenir prohibitif dans le cas de grandes bases.

Dans certaines approches, la réduction du nombre d'agrégats se fait uniquement par un arrêt prématuré de la recherche. Tous les agrégats sont d'abord ordonnés par un critère de proximité de la requête, puis on visite un nombre arbitraire d'agrégats. Ce genre d'approche n'offre aucun contrôle a priori de la qualité des résultats. C'est le cas de la méthode CLINDEX [LCGMW02], qui a, en revanche, l'avantage de proposer un

algorithme de regroupement de données très rapide. Lors de la recherche, les agrégats y sont ordonnés par la distance entre leur centre et la requête.

D'autres méthodes proposées récemment dans la littérature, utilisent une approche probabiliste pour l'étape de sélection des agrégats. Elles permettent d'effectuer un contrôle a priori de la précision des résultats obtenus dans le cas d'une recherche de plus proche voisin. La méthode DBIN (Density-based Indexing), propose un algorithme de regroupement des données basé sur une approche d'estimation de densité des données en utilisant un algorithme EM. La distribution des données est modélisée par un mélange de lois gaussiennes, chacune représentant un agrégat. L'algorithme EM est appliqué pour estimer les paramètres des lois gaussiennes. Lors de la recherche, les agrégats sont ordonnés par la probabilité que la requête appartienne à chacun d'entre eux. Les agrégats sont ensuite parcourus l'un après l'autre par ordre de probabilité. Les résultats de la recherche séquentielle de l'agrégat courant sont alors utilisés pour mettre à jour la probabilité qu'un vecteur soit meilleur que le plus proche voisin actuellement retrouvé. La recherche s'arrête lorsque cette probabilité est inférieure à une seuil fixé par l'utilisateur. L'inconvénient de cette méthode est lié directement à l'utilisation de l'algorithme EM. La complexité est assez élevée. Il y a en effet une limitation du nombre d'agrégats alors que certaines données réelles sont constituées d'un grand nombre d'agrégats de petite taille.

Berrani *et al.* ont proposé dans [BAG03] une méthode d'approximation avec contrôle de la précision. La méthode combine une approximation géométrique des hyper-sphères englobantes et le contrôle de la précision. Ce dernier se fait en fonction de la probabilité maximale de ne pas retrouver dans le résultat approximatif un vecteur qui serait présent dans le résultat exact. Le rayon des hyper-sphères englobantes est réduit en fonction du niveau d'imprécision fixé par l'utilisateur pour augmenter le nombre de clusters éliminés. Les expérimentations ont montré que la méthode proposée était 6.71 fois plus rapide qu'une recherche séquentielle avec une perte de 1% de précision. Cependant, la méthode a été testée avec des données de dimension moyenne (24 dimensions). Dans le cas de grande dimension, les vecteurs ont tendance à se concentrer à la surface des hyper-sphères. Par conséquent, la réduction des rayons n'est alors pas significative. De plus, les rayons réduits sont calculés hors-ligne d'après un taux d'imprécision fixé par l'utilisateur. Lorsque cette valeur change, il faut recalculer tous les rayons réduits.

2.4.4 Techniques basées sur l'algorithme MEDRANK

Fagin *et al.* ont proposé dans [FKS03] une méthode très rapide de recherche des plus proches voisins. La méthode est basée sur la projection aléatoire des données [Kle97] et un algorithme d'agrégation des *rangs* [DKNS01] appelé MEDRANK. L'étape de projection aléatoire des données consiste à sélectionner un nombre arbitraire de droites aléatoire, K ($K < D$, où D est le nombre de dimensions de données) et à projeter les données sur ces droites choisies. Les valeurs projetées sur chaque droite sont ordonnées. On obtient alors K listes ordonnées. Étant donnée une requête $\mathbf{r}$, elle est d'abord projetée sur les droites aléatoires pré-définies. Les coordonnées de $\mathbf{r}$ projetées sur chaque droite, r_i, sont localisées dans les listes correspondantes et indiquées par deux pointeurs

l_i et r_i pour chaque liste. La recherche parcourt les listes dans les deux directions en déplaçant toujours le pointeur plus proche de r_i. Le premier vecteur retrouvé dans plus de $K/2$ listes est considéré comme le plus proche voisin. Le deuxième vecteur retrouvé dans plus de $K/2$ listes est considéré comme le deuxième plus proche voisin et ainsi de suite. L'algorithme s'arrête lorsque le $k^{ème}$ plus proche voisin est retrouvé. Une alternative de MEDRANK est OMEDRANK qui, pour chaque étape du parcours, déplace toujours les deux pointeurs l_i et r_i sans déterminer lequel est plus proche de r_i. Ceci permet d'économiser une comparaison.

La méthode proposée a l'avantage d'examiner une petite portion de données seulement (par ex. 5%), et elle est très rapide. L'inconvénient majeur de cette approche est que la précision des résultats dépend du nombre de droites aléatoires utilisées, K, ainsi que de ces droites. Plus K est grand, meilleure est la précision, et plus l'algorithme est lent.

Pour améliorer la précision des résultats sans dégrader la performance de recherche, Lejsek *et al.* ont proposé dans [LA04, LAJA09] une méthode d'indexation, appelée PvS. La méthode consiste à créer des *PvS-indexes*. La construction d'un *PvS-index* prend d'abord une première droite aléatoire et projette les données sur cette droite (niveau 1). Les données sont partitionnées en segments de même taille (avec ou sans chevauchement). Ensuite, pour chaque segment, les données sont projetées une deuxième fois sur une autre droite aléatoire (niveau 2). La projection s'arrête quand toutes les droites aléatoires sont utilisées. Les valeurs projetées au dernier niveau sont stockées dans un B^+-tree. Les PvS-indexes ont donc une forme d'arbre. Dans la phase de recherche, pour chaque niveau, le segment dont le centre est plus proche de la valeur projetée de la requête est choisi. Cette étape se répète jusque au choix d'un segment au dernier niveau. Le B^+-tree associé à ce segment joue le même rôle qu'une liste ordonnée dans l'algorithme MEDRANK.

Intuitivement, la méthode d'indexation PvS réduit le temps de réponse et améliore la précision des résultats de l'algorithme MEDRANK (ou OMEDRANK) grâce à la reprojection des données. Cependant, le partitionnement des données en segments peut aussi séparer les vecteurs qui sont éloignés sur une droite mais sont proches sur d'autres droites. Lorsque la première droite est sélectionnée pour faire un partitionnement au premier niveau, les vecteurs séparés par un partitionnement n'ont plus de chance de se retrouver dans un même segment. Le *rappel* est ainsi dégradé.

La méthode d'indexation PvS a été appliquée dans le cas de détection de copie d'images [LÁJA05] en utilisant un algorithme de vote sur les descripteurs locaux où la recherche de plus proches d'un descripteur sert à la mise en correspondance des descripteurs locaux. La méthode ne nécessite pas tous les plus proches voisins d'un descripteur (c-à-d. le *rappel*) mais met l'accent sur la qualité des plus proches voisins (c-à-d. la *précision*) car l'étape de mise en correspondance des descripteurs locaux n'est qu'une étape intermédiaire.

2.5 Synthèse

Ce chapitre a dressé un bref aperçu des méthodes d'indexation pour accélérer la recherche des plus proches voisins d'une requête. D'abord, nous avons présenté des méthodes d'indexation multidimensionnelle, à savoir les techniques basées sur le partitionnement des données et les techniques de partitionnement de l'espace. La performance de ces méthodes se dégrade rapidement lorsque le nombre de dimensions augmente. C'est pour cela que des méthodes d'accélération de la recherche séquentielle ont été proposées. Nous avons ensuite présenté les techniques de recherche approximative qui réduisent le temps de réponse au détriment de la qualité des résultats. La première approche d'approximation de la recherche des plus proches voisins consiste à transformer la recherche en recherche à ε-près en estimant le rayon sur un échantillon de données. D'autres techniques visent à modéliser ou à représenter les données dans un autre espace plus petit que l'espace original. Il s'agit des méthodes de réduction de dimensions ou de modélisation de données comme l'ACP, la projection aléatoire. Les méthodes de réduction de dimensions accélèrent la recherche et améliorent aussi la qualité des résultats grâce au nettoyage du bruit. D'autres techniques consistent à réduire le nombre de vecteurs à comparer avec la requête y compris les méthodes d'approximation géométrique des requêtes, les méthodes basées sur le clustering et les méthodes basées sur l'agrégation des *rangs*. L'approximation géométrique des requêtes permet d'augmenter le nombre d'agrégats éliminés lors d'une étape de filtrage. La recherche est ainsi accélérée. Les méthodes basées sur l'agrégation des *rangs* comme MEDRANK et l'indexation PvS combinent la projection aléatoire et l'agrégation des *rangs* accélèrent considérablement la recherche.

Toutes les méthodes présentées dans ce chapitre ont chacune des avantages et des inconvénients. Le chapitre suivant est consacré à une approche originale qui combine plusieurs méthodes pour prendre des avantages de chacune : l'objectif est d'accélérer la recherche sans dégrader la qualité des résultats. L'approche s'appuie sur l'analyse factorielle des correspondances qui est développée originellement pour l'analyse des données textuelles lors que les documents sont représentés par des *sacs-de-mots*.

Chapitre 3

Analyse factorielle des correspondances pour l'indexation et la recherche d'images

3.1 Introduction

Le contenu de ce chapitre présente notre contribution au problème d'indexation et la recherche d'images par le contenu. Nous nous sommes inspirés des travaux en analyse de données textuelles et les avons adaptés à l'analyse des images. La méthode privilégiée est l'analyse factorielle des correspondances (AFC) qui traite des tableaux de contingence. Dans le cas des images, ces tableaux sont construits à partir des descripteurs locaux. Après avoir présenté la méthode AFC et son adaptation aux images, nous proposons une variante permettant d'accélérer la recherche des images. Nous terminons par des expérimentations, l'évaluation de notre méthode sur plusieurs bases d'images et la comparaison de notre méthode avec des méthodes de références comme la pondération *tf*idf*, le LSA et le PLSA.

3.2 Analyse factorielle des correspondances

L'Analyse factorielle des correspondances (AFC) a été proposée sous ce nom et développée par Benzécri [Ben73] dans le contexte de la linguistique, c'est-à-dire pour l'analyse des données textuelles (ADT). L'AFC est une méthode exploratoire adaptée aux *tableaux de contingence* et permet d'étudier les éventuelles relations existant entre deux variables nominales.

Le tableau de contingence (dit aussi de dépendance, ou tableau croisé) est obtenu en ventilant une population selon deux variables nominales. L'ensemble des lignes du tableau désigne les modalités d'une variable et l'ensemble des colonnes correspond à celles de l'autre variable. De ce fait, les lignes et les colonnes, qui désignent deux partitions d'une même population, jouent des rôles symétriques et sont traitées de façon analogue.

L'AFC sur un tableau croisant des documents et des mots permet de répondre aux questions suivantes : y a-t-il des proximités entre certains mots ? Idem pour les documents ? Existe-t-il des liens entre mots et documents ? L'AFC comme la plupart des méthodes factorielles utilise la décomposition en valeurs propres et vecteurs propres d'une matrice particulière [1] et permet la visualisation des mots et des documents dans un espace de dimension réduit. Cet espace de dimension réduit a la particularité d'avoir un nuage de points projetés (mots et/ou documents) d'inertie maximale. Par ailleurs, l'AFC fournit des indicateurs pertinents pour l'interprétation des axes comme la *contribution* d'un mot ou d'un document à l'inertie d'un axe ou la *qualité de représentation* d'un mot et/ou d'un document sur un axe [Mor04, Gre07].

3.2.1 Principe

Soit un tableau de contingence $\mathbf{X} = \{x_{ij}\}_{M,N}$ à M lignes et N colonnes obtenu en croisant M documents d'un corpus $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$ et N mots d'un vocabulaire $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$.

$\mathbf{X}$	v_1	v_2	$\ldots$	v_j	$\ldots$	v_N	Total
d_1	x_{11}	x_{12}	$\ldots$	x_{1j}	$\ldots$	x_{1N}	$x_{1.}$
d_2	x_{21}	x_{22}	$\ldots$	x_{2j}	$\ldots$	x_{2N}	$x_{2.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
d_i	x_{i1}	x_{i2}	$\ldots$	x_{ij}	$\ldots$	x_{iN}	$x_{i.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
d_M	x_{M1}	x_{M2}	$\ldots$	x_{Mj}	$\ldots$	x_{MN}	$x_{M.}$
Total	$x_{.1}$	$x_{.2}$	$\ldots$	$x_{.j}$	$\ldots$	$x_{.N}$	$x_{..}$

Chaque élément x_{ij} représente le nombre de fois où le mot j apparaît dans le document i. Le total marginal en lignes $x_{i.}$ est le nombre de mots (la longueur) du document i, alors que le total marginal $x_{.j}$ est le nombre de fois où le mot j apparaît dans le corpus.

On a les relations suivantes :

$$x_{i.} = \sum_{j=1}^{N} x_{ij} \tag{3.1}$$

$$x_{.j} = \sum_{i=1}^{M} x_{ij} \tag{3.2}$$

$$x_{..} = \sum_{i=1}^{M} \sum_{j=1}^{N} x_{ij} \tag{3.3}$$

qui en termes de *fréquences relatives*, donnent lieu aux relations :

1. L'AFC peut utiliser la décomposition en valeurs propres et vecteurs propres d'une matrice symétrique ou la décomposition en valeurs singulières (SVD) d'une matrice rectangulaire.

$$f_{ij} = \frac{x_{ij}}{x_{..}} \tag{3.4}$$

$$f_{i.} = \sum_{j=1}^{N} f_{ij} \tag{3.5}$$

$$f_{.j} = \sum_{i=1}^{M} f_{ij} \tag{3.6}$$

$$\sum_{i=1}^{M}\sum_{j=1}^{N} f_{ij} = 1 \tag{3.7}$$

où $f_{i.}$ est la *fréquence relative marginale* (ou la *masse*) du document i et $f_{.j}$ est la fréquence relative marginale du mot j.

On obtient alors le tableau $\mathbf{F} = \{f_{ij}\}_{M,N}$:

F	v_1	v_2	$\dots$	v_j	$\dots$	v_N	Total
d_1	f_{11}	f_{12}	$\dots$	f_{1j}	$\dots$	f_{1N}	$f_{1.}$
d_2	f_{21}	f_{22}	$\dots$	f_{2j}	$\dots$	f_{2N}	$f_{2.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
d_i	f_{i1}	f_{i2}	$\dots$	f_{ij}	$\dots$	f_{iN}	$f_{i.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
d_M	f_{M1}	f_{M2}	$\dots$	f_{Mj}	$\dots$	f_{MN}	$f_{M.}$
Total	$f_{.1}$	$f_{.2}$	$\dots$	$f_{.j}$	$\dots$	$f_{.N}$	1

$$\mathbf{F} = \frac{1}{x_{..}}\mathbf{X} \tag{3.8}$$

où $x_{..}$ est l'*effectif total* du tableau de contingence $\mathbf{X}$ obtenu par la formule 3.3. Par convention, nous appelons $\mathbf{X}$ le *tableau de contingence brut* et $\mathbf{F}$ le *tableau des fréquences relatives*.

Pour analyser un tableau de contingence, ce n'est pas le tableau d'effectifs bruts (le tableau de contingence brut $\mathbf{X}$ ou le tableau des fréquences relatives $\mathbf{F}$) qui nous intéresse mais les tableaux des *profils-lignes* et celui des *profils-colonnes* c'est-à-dire les répartitions en pourcentage à l'intérieur d'une ligne ou d'une colonne.

On note les profils-lignes :

$$\left(\frac{f_{ij}}{f_{i.}}\right)_{j=1..N} = \left(\frac{x_{ij}}{x_{i.}}\right)_{j=1..N} \tag{3.9}$$

et les profils-colonnes :

$$\left(\frac{f_{ij}}{f_{.j}}\right)_{i=1..M} = \left(\frac{x_{ij}}{x_{.j}}\right)_{i=1..M} \tag{3.10}$$

Le *profil-ligne moyen* s'exprime par :

$$(f_{.j})_{j=1..N} = \left(\frac{x_{.j}}{x_{..}}\right)_{j=1..N} \tag{3.11}$$

et le *profil-colonne moyen* est :

$$(f_{i.})_{i=1..M} = \left(\frac{x_{i.}}{x_{..}}\right)_{i=1..M} \tag{3.12}$$

Examiner les proximités entre les profils revient à examiner la proximité entre chaque profil et son profil moyen.

Nuage des M lignes

L'ensemble des profils-lignes forme un nuage de M points dans l'espace des N colonnes ($\mathbb{R}^N$). Chaque point i a pour coordonnées :

$$\left(\frac{f_{ij}}{f_{i.}}\right)_{j=1..N}$$

Il est affecté d'une masse $f_{i.}$ qui est sa fréquence relative. Le centre de gravité de ce nuage est la moyenne des profils-lignes affectés de leurs masses et correspond au profil moyen. Sa $j^{ème}$ composante vaut :

$$\sum_{i=1}^{M} f_{i.}\frac{f_{ij}}{f_{i.}} = f_{.j}$$

C'est la fréquence relative marginale des colonnes.

Nuage des N colonnes

De la même façon, l'ensemble des N profils-colonnes constitue un nuage de N points dans l'espace des M lignes ($\mathbb{R}^M$). Les coordonnées du point j sont données par :

$$\left(\frac{f_{ij}}{f_{.j}}\right)_{i=1..M}$$

Chaque point est affecté d'une masse $f_{.j}$. Le centre de gravité du nuage des profils-colonnes est le profil moyen. Sa $i^{ème}$ composante vaut :

$$\sum_{j=1}^{N} f_{.j}\frac{f_{ij}}{f_{.j}} = f_{i.}$$

C'est la fréquence relative marginale des lignes.

Critère d'ajustement

Nous voulons représenter graphiquement les proximités entre profils, ce qui nous conduit à représenter les proximités entre les profils et le profil moyen défini sur l'ensemble de la population. Dans la construction des nuages de $\mathbb{R}^N$ et de $\mathbb{R}^M$, le choix des profils comme coordonnées donne à tous les documents dans le corpus et tous les mots dans le vocabulaire la même importance. L'importance est cependant restituée au travers de la masse affectée à chaque point (proportionnelle à sa fréquence), afin de ne pas privilégier les classes d'effectifs faibles et de respecter la répartition réelle de la population. Cette masse interviendra d'une part lors du calcul des coordonnées du centre de gravité du nuage et d'autre part dans le critère d'ajustement. Pour le calcul de l'ajustement, la quantité à maximiser sera donc la somme pondérée des carrés des distances entre les points et le centre de gravité du nuage (c'est-à-dire l'inertie de la droite d'allongement maximum du nuage).

Choix des distances

La distance euclidienne usuelle entre deux points-lignes exprimée sur le tableau d'effectifs bruts ne ferait que traduire les différences d'effectifs entre deux documents. En revanche la distance euclidienne usuelle entre deux profils-lignes traduit bien la ressemblance ou la différence entre les deux documents sans tenir compte des effectifs totaux (les longueurs dans ce cas) de ces documents :

$$d^2(i,i') = \sum_{j=1}^{N} \left(\frac{f_{ij}}{f_{i.}} - \frac{f_{i'j}}{f_{i'.}} \right)^2 \tag{3.13}$$

Cette distance favorise les colonnes qui ont une masse $f_{.j}$ importante. Pour remédier à ce problème, et aussi pour conserver la symétrie des lignes et colonnes, on pondère chaque écart par l'inverse de la masse de la colonne et l'on calcule une nouvelle distance appelée [2] *la distance du* χ^2 :

$$d^2(i,i') = \sum_{j=1}^{N} \frac{1}{f_{.j}} \left(\frac{f_{ij}}{f_{i.}} - \frac{f_{i'j}}{f_{i'.}} \right)^2 \tag{3.14}$$

On définit de la même manière la distance entre les profils-colonnes par :

$$d^2(j,j') = \sum_{i=1}^{M} \frac{1}{f_{i.}} \left(\frac{f_{ij}}{f_{.j}} - \frac{f_{ij'}}{f_{.j'}} \right)^2 \tag{3.15}$$

C'est cette distance pondérée, ainsi que le rôle symétrique joué par les lignes et les colonnes du tableau de contingence, qui particularisent l'analyse factorielle des correspondances et lui assurent propriétés remarquables que ne possèdes pas l'analyse en composantes principales : l'équivalence distributionnelle et les relations de transition.

2. L'inertie totale des nuages de points-lignes (ou de points-colonnes) calculée avec cette distance est proportionnelle au classique χ^2 de Karl Pearson utilisé pour éprouver l'indépendance des lignes et des colonnes d'un tableau de contingence. D'où le nom de distance du χ^2.

3.2.2 Schéma général de l'analyse factorielle des correspondances

L'analyse factorielle des correspondances revient à effectuer l'analyse générale d'un nuage de points pondérés dans un espace muni de la métrique χ^2.

Éléments de base

Les transformations opérées sur le tableau des données peuvent s'écrire à partir de trois matrices, $\mathbf{F}$, $\mathbf{P}$ et $\mathbf{Q}$ qui définissent les éléments de base de l'analyse.

- $\mathbf{F}$ de dimension $M \times N$ désigne le tableau des fréquences relatives ;
- $\mathbf{P}$ est la matrice diagonale de dimension $M \times M$ dont les éléments diagonaux sont les fréquences relatives marginales des lignes $f_{i.}$;
- $\mathbf{Q}$ de dimension $N \times N$ est la matrice diagonale dont les éléments diagonaux sont les fréquences marginales des colonnes $f_{.j}$.

Le tableau de profils-lignes s'exprime alors sous forme matricielle par $\mathbf{P}^{-1}\mathbf{F}$ et celui des profils-colonnes $\mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}$.

Critère à maximiser et matrice à diagonaliser

Nous nous plaçons, dans les deux espaces, aux centres de gravité des nuages. Cependant il est équivalent de procéder à l'analyse par rapport à l'origine ou par rapport aux centres de gravité, à condition de négliger dans le premier cas l'axe factoriel qui joint l'origine au centre de gravité [3].

Nous effectuons ci-dessous l'analyse par rapport à l'origine car l'expression des formules est plus simple. Plaçons-nous dans l'espace des colonnes $\mathbb{R}^N$. Nous faisons une analyse générale du tableau des profils-lignes $\mathbf{P}^{-1}\mathbf{F}$ avec la métrique $\mathbf{Q}^{-1}$ (la métrique χ^2) et la matrice des masses $\mathbf{P}$. En effet, nous cherchons l'axe d'inertie maximum du nuage des points-lignes (profils-lignes) passant par l'origine O et engendré par un vecteur-unitaire $\mathbf{u}$ pour la métrique $\mathbf{Q}^{-1}$.

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} \quad \text{et} \quad \mathbf{u}^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{u} = 1$$

Ceci nous amène à maximiser la somme pondérée des carrés des projections sur l'axe :

$$\max_{\mathbf{u}} \left\{ \sum_{i=1}^{M} f_{i.} d^2_{\mathbf{u}}(i, O) \right\}$$

où $d^2_{\mathbf{u}}(i, O)$ est le carré de la projection du point i (profil-ligne i) sur l'axe engendré par $\mathbf{u}$ (pour la métrique $\mathbf{Q}^{-1}$).

3. Cet axe est associé à la valeur propre égale à 1, appelée valeur propre triviale

Cela revient à maximiser la quantité :

$$\begin{aligned} \textit{inertie} &= \mathbf{u}^{\mathrm{T}}\mathbf{Q}^{-1}(\mathbf{P}^{-1}\mathbf{F})^{\mathrm{T}}\mathbf{P}(\mathbf{P}^{-1}\mathbf{F})\mathbf{Q}^{-1}\mathbf{u} \\ &= \mathbf{u}^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}\mathbf{u} \end{aligned} \tag{3.16}$$

avec la contrainte :

$$\mathbf{u}^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{u} = 1 \tag{3.17}$$

$\mathbf{u}$ est vecteur propre de la matrice :

$$\mathbf{A} = \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1} \tag{3.18}$$

associé à la plus grande valeur propre λ différente de 1.

De la même façon, on doit rendre maximum dans $\mathbb{R}^M$, la quantité :

$$\mathbf{v}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{v} \tag{3.19}$$

avec la contrainte :

$$\mathbf{v}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{v} = 1 \tag{3.20}$$

$\mathbf{v}$ est vecteur propre de la matrice :

$$\mathbf{B} = \mathbf{F}\mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1} \tag{3.21}$$

Axes factoriels

Nous supposons ici que N correspond à la plus petite dimension du tableau de données. La matrice à diagonaliser $\mathbf{A} = \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}$ a au plus N valeurs propres non nulles :

$$\boldsymbol{\lambda} = \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{N-1} \end{pmatrix} \quad \text{où} \quad 1 = \lambda_0 \geq \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N-1} \geq 0$$

et N vecteurs propres associés $(\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{N-1})$ avec

$$\mathbf{u}_0 = \begin{pmatrix} f_{.1} \\ f_{.2} \\ \vdots \\ f_{.N} \end{pmatrix}$$

Après avoir écarté la valeur propre triviale λ_0 qui est égale à 1 et le vecteur propre associé $\mathbf{u}_0$, nous retenons, de la diagonalisation de la matrice, au plus $(N-1)$ valeurs propres non nulles et les vecteurs propres associés. Nous obtenons ainsi au plus $(N-1)$ axes factoriels. En l'analyse des données textuelles, on ne conserve le plus souvent que K (par ex. 30) axes associés aux plus grandes valeurs propres après avoir supprimé les $(N-K-1)$ plus petites valeurs propres qui sont très faibles et jugées "négligeables".

La projection des documents sur l'axe α (l'axe engendré par le vecteur $\mathbf{u}_\alpha$) est obtenu en projetant les profils-lignes sur cet axe avec la métrique $\mathbf{Q}^{-1}$:

$$\begin{aligned} \mathbf{z}_\alpha &= \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}\mathbf{u}_\alpha \\ &= \begin{pmatrix} z_{1\alpha} \\ z_{2\alpha} \\ \vdots \\ z_{M\alpha} \end{pmatrix} \end{aligned} \tag{3.22}$$

où $\mathbf{u}_\alpha$ est le α^e vecteur propre de la matrice $\mathbf{A}$ et $z_{i\alpha}$ $(1 \leq i \leq M)$ est la projection sur l'axe α (ou la *coordonnée factorielle*) du document i.

La nouvelle représentation des documents sur K axes factoriels, notée $\mathbf{Z}$, s'obtient alors par la juxtaposition de leurs projections sur chaque axe factoriel.

$$\begin{aligned} \mathbf{Z} &= \begin{pmatrix} z_{11} & z_{12} & \dots & z_{1K} \\ z_{21} & z_{22} & \dots & z_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ z_{M1} & z_{M2} & \dots & z_{MK} \end{pmatrix} \\ &= [\mathbf{z}_1\ \mathbf{z}_2\ \dots\ \mathbf{z}_K] \\ &= \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}[\mathbf{u}_1\ \mathbf{u}_2\ \dots\ \mathbf{u}_K] \\ &= \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}\mathbf{U} \end{aligned} \tag{3.23}$$

où $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \dots\ \mathbf{u}_K]$ est la matrice obtenue en juxtaposant les K vecteurs propres de la matrice $\mathbf{A}$.

De la même manière, la projection des mots sur l'axe α est calculée par :

$$\begin{aligned} \mathbf{w}_\alpha &= \mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{v}_\alpha \\ &= \begin{pmatrix} w_{1\alpha} \\ w_{2\alpha} \\ \vdots \\ w_{N\alpha} \end{pmatrix} \end{aligned} \tag{3.24}$$

et on obtient la nouvelle représentation des mots :

$$\begin{aligned} \mathbf{W} &= \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1K} \\ w_{21} & w_{22} & \dots & w_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \dots & w_{NK} \end{pmatrix} \\ &= \mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{V} \end{aligned} \tag{3.25}$$

où $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \dots\ \mathbf{v}_K]$ est la matrice des vecteurs propres de la matrice $\mathbf{B}$.

Les *coordonnées factorielles* (projections des points sur les axes factoriels) sont centrées :

$$\sum_{i=1}^{M} f_{i.} z_{i\alpha} = \sum_{j=1}^{N} f_{.j} w_{j\alpha} = 0 \tag{3.26}$$

et de variance égale à λ_α :

$$\sum_{i=1}^{M} f_{i.} z_{i\alpha}^2 = \sum_{j=1}^{N} f_{.j} w_{j\alpha}^2 = \lambda_\alpha \tag{3.27}$$

où $z_{i\alpha}$ est la coordonnée du point-document i sur l'axe α et $w_{j\alpha}$ est la coordonnée du point-mot j sur l'axe α.

Les éléments de construction de l'analyse sont récapitulés dans le tableau 3.1.

TABLE 3.1 – Éléments de construction de l'analyse

Dans $\mathbb{R}^N$	$\leftarrow$ Éléments de construction $\rightarrow$	Dans $\mathbb{R}^M$
$\mathbf{A} = \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}$	Matrice à diagonaliser	$\mathbf{B} = \mathbf{F}\mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}$
$\mathbf{A}\mathbf{u}_\alpha = \lambda_\alpha \mathbf{u}_\alpha$ $(\mathbf{u}_\alpha^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{u}_\alpha = 1)$	Axe factoriel	$\mathbf{B}\mathbf{v}_\alpha = \lambda_\alpha \mathbf{v}_\alpha$ $(\mathbf{v}_\alpha^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{v}_\alpha = 1)$
$\mathbf{z}_\alpha = \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}\mathbf{u}_\alpha$ $\mathbf{Z} = \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}\mathbf{U}$	Coordonnées factorielles	$\mathbf{w}_\alpha = \mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{v}_\alpha$ $\mathbf{W} = \mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{V}$

Relation entre les deux espaces

On montre que les matrices $\mathbf{A}$ et $\mathbf{B}$ ont les même valeurs propres non nulles λ_α et qu'entre le vecteur propre unitaire $\mathbf{u}_\alpha$ de $\mathbf{A}$ associé à la valeur propre λ_α et le vecteur propre unitaire $\mathbf{v}_\alpha$ de $\mathbf{B}$ relatif à la même valeur propre, il existe les relations de transition :

$$\mathbf{v}_\alpha = \frac{1}{\sqrt{\lambda_\alpha}} \mathbf{F}\mathbf{Q}^{-1}\mathbf{u}_\alpha \tag{3.28}$$

$$\mathbf{u}_\alpha = \frac{1}{\sqrt{\lambda_\alpha}} \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{v}_\alpha \tag{3.29}$$

ou :

$$\mathbf{V} = \mathbf{F}\mathbf{Q}^{-1}\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{3.30}$$

$$\mathbf{U} = \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{3.31}$$

où

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_K \end{pmatrix}$$

La comparaison de ces relations avec les expressions des coordonnées factorielles (cf. formules 3.22 et 3.24) montre que celles-ci sont liées aux composantes des axes de l'autre espace par les formules :

$$\mathbf{z}_\alpha = \sqrt{\lambda_\alpha}\mathbf{P}^{-1}\mathbf{v}_\alpha \tag{3.32}$$
$$\mathbf{w}_\alpha = \sqrt{\lambda_\alpha}\mathbf{Q}^{-1}\mathbf{u}_\alpha \tag{3.33}$$

ou :

$$\mathbf{Z} = \mathbf{P}^{-1}\mathbf{V}\mathbf{\Lambda}^{\frac{1}{2}} \tag{3.34}$$
$$\mathbf{W} = \mathbf{Q}^{-1}\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}} \tag{3.35}$$

Relation de transition

Les substitutions dans la relation (3.24) de $\mathbf{v}_\alpha$ par sa valeur tirée de (3.32) et dans la relation (3.22) de $\mathbf{u}_\alpha$ par sa valeur tirée de (3.33) conduisent aux relations fondamentales existant entre les coordonnées des points-lignes et des points-colonnes sur l'axe α, les relations quasi-barycentriques :

$$\mathbf{z}_\alpha = \frac{1}{\sqrt{\lambda_\alpha}}\mathbf{P}^{-1}\mathbf{F}\mathbf{w}_\alpha \tag{3.36}$$
$$\mathbf{w}_\alpha = \frac{1}{\sqrt{\lambda_\alpha}}\mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{z}_\alpha \tag{3.37}$$

qui peuvent s'exprimer sous forme matricielle :

$$\mathbf{Z} = \mathbf{P}^{-1}\mathbf{F}\mathbf{W}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{3.38}$$
$$\mathbf{W} = \mathbf{Q}^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{Z}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{3.39}$$

Ainsi, au coefficient de dilatation $\frac{1}{\sqrt{\lambda_\alpha}}$ près, les projections des points représentatifs d'un nuage sont, sur un axe, les *barycentres* des projections des points représentatifs de l'autre nuage.

Reconstitution des données

Pour des vecteurs de projection $\mathbf{z}_\alpha, \mathbf{w}_\alpha$ normés à 1, le tableau des fréquences relatives est reconstitué exactement par :

$$f_{ij} = f_{i.}f_{.j}\sum_{\alpha=0}^{N-1}\sqrt{\lambda_\alpha}z_{i\alpha}w_{j\alpha} \tag{3.40}$$

Si les $(N-K-1)$ plus petites valeurs propres sont très faibles et jugées "négligeables", on peut limiter la sommation aux K premiers termes correspondant aux valeurs propres $(\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_K)$:

$$f_{ij} \approx f_{i.} f_{.j} \sum_{\alpha=0}^{K} \sqrt{\lambda_\alpha} z_{i\alpha} w_{j\alpha} \tag{3.41}$$

Notons que, dans les formules de reconstitution de données, nous devons prendre aussi l'axe trivial dont la valeur propre λ_0 est égale à 1.

Éléments supplémentaires

Pour un nouveau document (par ex. une requête) $\mathbf{r} = (r_1\ r_2\ \ldots\ r_N)$, il s'agit de situer le nouveau point-ligne par rapport au M points-lignes analysés. Soit r_j la $j^{ème}$ coordonnée du document supplémentaire. Son profil est donné par le vecteur ligne $\hat{\mathbf{r}}$:

$$\hat{\mathbf{r}} = \left(\hat{r}_j = \frac{r_j}{r_.} \right)_{j=1..N} \quad \text{avec} \quad r_. = \sum_{j=1}^{N} r_j$$

On projette ce point sur l'axe α en utilisant la même formule de transition pour les lignes du tableau des fréquences relatives (cf. Formule 3.36) :

$$\begin{aligned} z_\alpha^{(\mathbf{r})} &= \frac{1}{\sqrt{\lambda_\alpha}} \hat{\mathbf{r}}\, \mathbf{w}_\alpha \\ &= \frac{1}{\sqrt{\lambda_\alpha}} \sum_{j=1}^{N} \hat{r}_j w_{j\alpha} \end{aligned} \tag{3.42}$$

À l'instar des éléments analysés, les éléments supplémentaires se calculent et s'interprètent comme des quasi-barycentres.

La représentation du nouveau document sur K axes factoriels est donc donnée par :

$$\begin{aligned} \mathbf{z}^{(\mathbf{r})} &= \left[z_1^{(\mathbf{r})}\ z_2^{(\mathbf{r})}\ \ldots\ z_K^{(\mathbf{r})} \right] \\ &= \hat{\mathbf{r}}\, \mathbf{W} \Lambda^{-\frac{1}{2}} \end{aligned} \tag{3.43}$$

Notons bien que les coordonnées factorielles du nouveau document $\mathbf{r}$ peuvent être calculées directement à partir des vecteurs propres $\mathbf{u}_\alpha$ (cf. Formule 3.22) :

$$z_\alpha^{(\mathbf{r})} = \hat{\mathbf{r}}\, \mathbf{Q}^{-1} \mathbf{u}_\alpha \tag{3.44}$$

et que sa représentation sur K axes est alors :

$$\mathbf{z}^{(\mathbf{r})} = \hat{\mathbf{r}}\, \mathbf{Q}^{-1} \mathbf{U} \tag{3.45}$$

3.2.3 Mise en œuvre des calculs

Grâce aux relations entre les deux espaces $\mathbb{R}^N$ et $\mathbb{R}^M$, il est possible d'effectuer les calculs sur un seul espace. Les coordonnées factorielles de l'autre espace peuvent s'en déduire. Par ailleurs, la diagonalisation d'une petite matrice sera plus rapide que celle d'une grande matrice. Supposons que $M > N$, nous nous plaçons dans l'espace $\mathbb{R}^N$. L'analyse consiste à construire la matrice $\mathbf{A}$ et à calculer ses valeurs propres et les vecteurs propres associés.

Symétrisation de la matrice à diagonaliser

La matrice à diagonaliser $\mathbf{A} = \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}$, dans $\mathbb{R}^N$, n'est en général pas symétrique. En pratique, la diagonalisation d'une matrice symétrique est beaucoup plus rapide que celle d'une matrice non symétrique. C'est pour cela que nous cherchons à symétriser la matrice à diagonaliser.

Considérons la matrice $\mathbf{S}_0 = \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}$ symétrique et la matrice $\mathbf{Q}^{-1}$ diagonale. On s'exprime alors $\mathbf{A}$ par :

$$\begin{aligned} \mathbf{A} &= \mathbf{S}_0\mathbf{Q}^{-1} \\ &= \mathbf{S}_0\mathbf{Q}^{-\frac{1}{2}}\mathbf{Q}^{-\frac{1}{2}} \end{aligned}$$

Partant de la relation $\mathbf{A} = \lambda\mathbf{u}$, il vient :

$$\mathbf{S}_0\mathbf{Q}^{-\frac{1}{2}}\mathbf{Q}^{-\frac{1}{2}}\mathbf{u} = \lambda\mathbf{u}$$

Prémultiplions les deux membres par $\mathbf{Q}^{-\frac{1}{2}}$ et en posant $\mathbf{Q}^{-\frac{1}{2}}\mathbf{u} = \mathbf{t}$, nous obtenons :

$$\mathbf{Q}^{-\frac{1}{2}}\mathbf{S}_0\mathbf{Q}^{-\frac{1}{2}}\mathbf{t} = \lambda\mathbf{t}$$

La matrice

$$\begin{aligned} \mathbf{S} &= \mathbf{Q}^{-\frac{1}{2}}\mathbf{S}_0\mathbf{Q}^{-\frac{1}{2}} \\ &= \mathbf{Q}^{-\frac{1}{2}}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-\frac{1}{2}} \end{aligned} \tag{3.46}$$

est symétrique et

$$\mathbf{S}\mathbf{t} = \lambda\mathbf{t}$$

Les matrice $\mathbf{A}$ et $\mathbf{S}$ ont mêmes valeurs propres λ. Leurs vecteurs propres sont liés par la relation :

$$\mathbf{u} = \mathbf{Q}^{\frac{1}{2}}\mathbf{t} \tag{3.47}$$

La projection des documents sur K axes est alors obtenue par :

$$\begin{aligned} \mathbf{Z} &= \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}\mathbf{U} \\ &= \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-\frac{1}{2}}\mathbf{T} \end{aligned} \tag{3.48}$$

où $\mathbf{T} = [\mathbf{t}_1\ \mathbf{t}_2\ \ldots\ \mathbf{t}_K]$, et $\mathbf{U} = \mathbf{Q}^{\frac{1}{2}}\mathbf{T}$.

La projection des mots est donc (cf. Formule 3.35) :

$$\begin{aligned} \mathbf{W} &= \mathbf{Q}^{-1}\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}} \\ &= \mathbf{Q}^{-\frac{1}{2}}\mathbf{T}\mathbf{\Lambda}^{\frac{1}{2}} \end{aligned} \tag{3.49}$$

Calculs directs à partir du tableau de contingence brut

Les calculs montrés jusqu'à présent se basent sur le tableau des fréquences relatives $\mathbf{F}$ mais non pas sur le tableau de contingence brut. Pour raisons de commodité [4], il vaut la peine d'effectuer directement les calculs à partir du tableau de contingence brut $\mathbf{X}$.

Rappelons d'abord la relation entre $\mathbf{F}$ et $\mathbf{X}$:

$$\mathbf{F} = \frac{1}{x_{..}}\mathbf{X}$$

Nous calculons ensuite les matrices diagonales $\hat{\mathbf{P}}$ et $\hat{\mathbf{Q}}$ dont les éléments diagonaux sont des totaux marginaux en lignes $x_{i.}$ et en colonnes $x_{.j}$ respectivement. Nous avons les relations suivantes :

$$\mathbf{P} = \frac{1}{x_{..}}\hat{\mathbf{P}} \tag{3.50}$$

$$\mathbf{Q} = \frac{1}{x_{..}}\hat{\mathbf{Q}} \tag{3.51}$$

La matrice à diagonaliser $\mathbf{S}$ se réécrit de manière suivante :

$$\begin{aligned} \mathbf{S} &= \mathbf{Q}^{-\frac{1}{2}}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-\frac{1}{2}} \\ &= \left(\frac{1}{x_{..}}\right)^{-\frac{1}{2}+1-1+1-\frac{1}{2}} \hat{\mathbf{Q}}^{-\frac{1}{2}}\mathbf{X}^{\mathrm{T}}\hat{\mathbf{P}}^{-1}\mathbf{X}\hat{\mathbf{Q}}^{-\frac{1}{2}} \\ &= \hat{\mathbf{Q}}^{-\frac{1}{2}}\mathbf{X}^{\mathrm{T}}\hat{\mathbf{P}}^{-1}\mathbf{X}\hat{\mathbf{Q}}^{-\frac{1}{2}} \end{aligned} \tag{3.52}$$

La projection des documents se fait donc par :

$$\begin{aligned} \mathbf{Z} &= \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}\mathbf{U} \\ &= \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-\frac{1}{2}}\mathbf{T} \\ &= \left(\frac{1}{x_{..}}\right)^{-\frac{1}{2}} \hat{\mathbf{P}}^{-1}\mathbf{X}\hat{\mathbf{Q}}^{-\frac{1}{2}}\mathbf{T} \end{aligned} \tag{3.53}$$

et la projection des mots est alors :

$$\begin{aligned} \mathbf{W} &= \mathbf{Q}^{-1}\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}} \\ &= \mathbf{Q}^{-\frac{1}{2}}\mathbf{T}\mathbf{\Lambda}^{\frac{1}{2}} \\ &= \left(\frac{1}{x_{..}}\right)^{-\frac{1}{2}} \hat{\mathbf{Q}}^{-\frac{1}{2}}\mathbf{T}\mathbf{\Lambda}^{\frac{1}{2}} \end{aligned} \tag{3.54}$$

4. Par exemple, dans la construction incrémentale de la matrice $\mathbf{S}$, l'effectif total $x_{..}$ ne peut être connu qu'après le chargement du tableau entier. Il est donc impossible de définir le tableau des fréquences relatives $\mathbf{F}$ si on n'a pas $x_{..}$.

3.3 Exemple

Pour faciliter la compréhension des calculs de l'AFC, nous montrons ici un petit exemple. Considérons un tableau de contingence $\mathbf{X}$ obtenu en croisant un corpus de 4 documents $\mathcal{D} = \{d_1, d_2, d_3, d_4\}$ et un vocabulaire de 3 mots $\mathcal{V} = \{v_1, v_2, v_3\}$.

$\mathbf{X}$	v_1	v_2	v_3	Total
d_1	13	2	5	20
d_2	20	2	8	30
d_3	10	5	5	20
d_4	7	1	22	30
Total	50	10	40	100

Pour obtenir le tableau des fréquences relatives $\mathbf{F}$, on divise les éléments du tableau $\mathbf{X}$ par l'effectif total $x_{..}$ qui est égal à 100.

$\mathbf{F}$	v_1	v_2	v_3	Total
d_1	0.13	0.02	0.05	0.20
d_2	0.20	0.02	0.08	0.30
d_3	0.10	0.05	0.05	0.20
d_4	0.07	0.01	0.22	0.30
Total	0.50	0.10	0.40	1

Les matrices diagonales des fréquences relatives marginales $\mathbf{P}$ et $\mathbf{Q}$ sont :

$$\mathbf{P} = \begin{pmatrix} 0.2 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.3 \end{pmatrix} \quad \text{et} \quad \mathbf{Q} = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.4 \end{pmatrix}$$

On calcule ensuite le tableau des profils-lignes (correspondant aux documents) :

$$\mathbf{P}^{-1}\mathbf{F} = \begin{pmatrix} 0.6500 & 0.1000 & 0.2500 \\ 0.6667 & 0.0667 & 0.2667 \\ 0.5000 & 0.2500 & 0.2500 \\ 0.2333 & 0.0333 & 0.7333 \end{pmatrix}$$

La matrice symétrique à diagonaliser $\mathbf{S} = \mathbf{Q}^{-\frac{1}{2}}\mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-\frac{1}{2}}$ est :

$$\mathbf{S} = \begin{pmatrix} 0.5683 & 0.2400 & 0.3626 \\ 0.2400 & 0.1617 & 0.1508 \\ 0.3626 & 0.1508 & 0.5192 \end{pmatrix}$$

La matrice $\mathbf{S}$ a 3 valeurs propres :

$$\boldsymbol{\lambda} = \begin{pmatrix} 1.0000 \\ 0.1998 \\ 0.0494 \end{pmatrix}$$

et 3 vecteurs propres associés :

$$[\mathbf{t}_0\ \mathbf{t}_1\ \mathbf{t}_2] = \begin{pmatrix} -0.7071 & -0.5387 & 0.4581 \\ -0.3162 & -0.3385 & -0.8862 \\ -0.6325 & 0.7715 & -0.0690 \end{pmatrix}$$

Après écarté la première valeur propre (la valeur propre triviale qui est égale à 1) et le vecteur propre associé (la première colonne), on obtient alors 2 axes factoriels correspondant aux valeurs propres et valeurs propres suivants :

$$\mathbf{\Lambda} = \begin{pmatrix} 0.1998 & 0 \\ 0 & 0.0494 \end{pmatrix} \quad \text{et} \quad \mathbf{T} = \begin{pmatrix} -0.5387 & 0.4581 \\ -0.3385 & -0.8862 \\ 0.7715 & -0.0690 \end{pmatrix}$$

On projette les documents sur les 2 axes factoriels (cf. Formule 3.48) :

$$\begin{aligned} \mathbf{Z} &= \mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-\frac{1}{2}}\mathbf{T} \\ &= \mathbf{P}^{-1}\mathbf{F} \begin{pmatrix} 1.4142 & 0 & 0 \\ 0 & 3.1623 & 0 \\ 0 & 0 & 1.5811 \end{pmatrix} \begin{pmatrix} -0.5387 & 0.4581 \\ -0.3385 & -0.8862 \\ 0.7715 & -0.0690 \end{pmatrix} \\ &= \begin{pmatrix} 0.6500 & 0.1000 & 0.2500 \\ 0.6667 & 0.0667 & 0.2667 \\ 0.5000 & 0.2500 & 0.2500 \\ 0.2333 & 0.0333 & 0.7333 \end{pmatrix} \begin{pmatrix} -0.7618 & 0.6478 \\ -1.0706 & -2.8025 \\ 1.2199 & -0.1092 \end{pmatrix} \\ &= \begin{pmatrix} -0.2972 & 0.1135 \\ -0.2539 & 0.2159 \\ -0.3436 & -0.4040 \\ 0.6811 & -0.0223 \end{pmatrix} \end{aligned}$$

La projection des mots est obtenue par la relation entre deux espaces (cf. Formule 3.49).

$$\begin{aligned} \mathbf{W} &= \mathbf{Q}^{-\frac{1}{2}}\mathbf{T}\mathbf{\Lambda}^{\frac{1}{2}} \\ &= \begin{pmatrix} -0.7618 & 0.6478 \\ -1.0706 & -2.8025 \\ 1.2199 & -0.1092 \end{pmatrix} \begin{pmatrix} 0.4470 & 0 \\ 0 & 0.2222 \end{pmatrix} \\ &= \begin{pmatrix} -0.3405 & 0.1439 \\ -0.4785 & -0.6226 \\ 0.5453 & -0.0243 \end{pmatrix} \end{aligned}$$

Nous pouvons visualiser les documents et les mots sur un même plan factoriel (Figure 3.1). Dans ce graphique, nous voyons que les documents d_1 et d_2 sont "proches" du mot v_1 et sont très "loin" des mots v_3 et v_2. Cela signifie qu'il y a une forte relation entre les documents d_1, d_2 et le mot v_1 et que le mot v_1 représente bien les documents d_1, d_2.

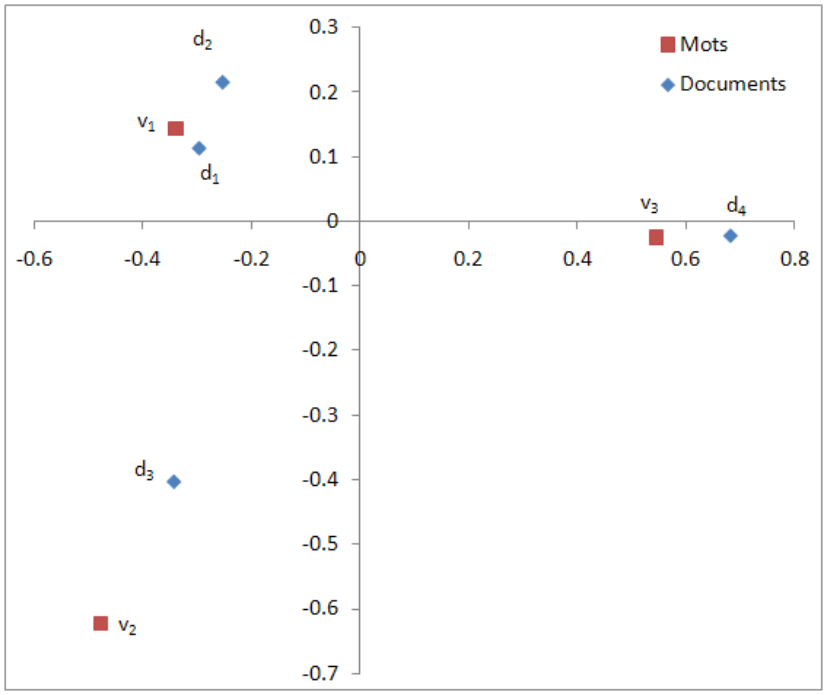


FIGURE 3.1 – Représentation simultanée des documents et des mots sur un même plan factoriel

3.4 Adaptation de l'AFC pour la recherche d'images

Puisque l'AFC ne traite que des tableaux de contingence, pour l'adaptation aux images, nous devons représenter les images sous forme d'un tableau de contingence. Nous utilisons les « *résumés des descripteurs locaux* » décrits dans la section 1.2.3 du chapitre 1 pour représenter les images. Le tableau de contingence est obtenu en croisant les *images* et les *mots visuels*.

Après obtention du tableau de contingence, l'application de l'AFC aux images est analogue aux textes. Dans ce cas, nous avons les *images* comme *documents* et les *mots visules* comme *mots*.

3.4.1 Construction des mots visuels

Les mots dans les images, appelés *mots visuels*, sont calculés à partir des descripteurs locaux pour constituer un vocabulaire de N *mots visuels*. La construction des mots visuels se fait donc en deux étapes : (i) calcul des descripteurs locaux pour un ensemble d'images, (ii) classification des descripteurs calculés en N clusters. Chaque *cluster* sera, par définition, un *mot visuel*.

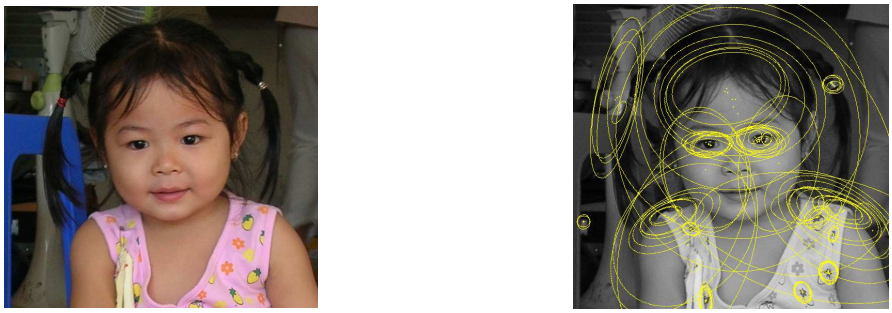


FIGURE 3.2 – Points d'intérêt détectés par un détecteur Hessian-Affine.

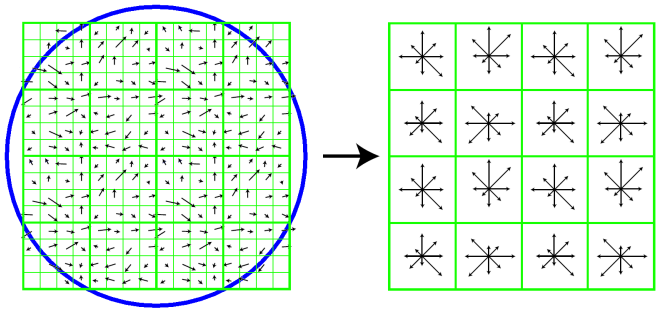


FIGURE 3.3 – Un descripteur SIFT calculé à partir de la région autour d'un point d'intérêt (le cercle) : gradient de l'image (à gauche), et le descripteur du point d'intérêt (à droite).

Description locale par des points d'intérêt

Le calcul des descripteurs locaux dans une image consiste d'abord à détecter des points d'intérêt et à caractériser ces points (cf. Section 1.2.2). La figure 3.2 montre quelques points d'intérêt détectés par un détecteur Hessien-Affine [MS02]. La description des points d'intérêt se fait sur la région autour de ces points. Nous avons utilisé le descripteur SIFT [Low04] pour caractériser les points d'intérêt car il est le plus utilisé des descripteurs proposés dans la littérature. Chaque descripteur SIFT est un vecteur à 128 dimensions. Le calcul des descripteurs SIFT est illustré sur la figure 3.3.

Classification des descripteurs locaux

La seconde étape de la construction des mots visuels consiste à former des mots visuels à partir des descripteurs locaux obtenus à l'étape précédente. La plupart des travaux effectuent un k-means sur les descripteurs locaux et prennent les centres de gravité de chaque *cluster* comme *mots visuels* [SRE+05, BZM06]. Chaque descripteur est ensuite affecté au *cluster* le plus proche et il est étiqueté par le *mot visuel* associé au *cluster* correspondant. Il s'agit d'une quantification vectorielle des descripteurs locaux.

3.4.2 Construction du tableau de contingence

Après avoir construit le vocabulaire visuel et affecté les descripteurs aux mots visuels correspondant, une image est caractérisée par la distribution des mots visuels (la fréquence des mots visuels qui apparaissent dans l'image). On obtient ainsi le tableau de contingence pour les images.

D'autres codages sont envisageables comme par exemple les *pixmots*. Les *pixmots* ont été introduits par M. Kerbaol et utilisés par F. Le Guillarm pour un projet informatique d'encodage d'images noir et blanc [LG07].

3.4.3 Amélioration de la recherche d'images par l'AFC

La première approche, pour la recherche d'images inspirée de la recherche de textes utilise un modèle de sac-de-mots-visuels [SZ03b]. Les deux modèles : sac-de-mots-textuels et sac-de-mots-visuels se basent sur le modèle vectoriel (*vector space model* [SWY75]) de la recherche d'information et partagent une même forme de représentation : la matrice de co-occurrences ou le tableau de contingence. Par conséquent, les méthodes basées sur les textes peuvent être utilisées pour améliorer la recherche d'images. Cependant, il existe certaines différences fondamentales entre les deux modèles : la sémantique des mots visuels, et la nature des requêtes. En fait, un mot textuel décrit généralement un concept (par ex. voiture, maison), tandis qu'un mot visuel correspond à une partie d'un objet. Donc, plusieurs mots visuels sont nécessaires pour décrire un objet. Cette observation montre qu'il y a encore un *fossé sémantique* entre les mots visuels et les objets dans la description sémantique du contenu des images [5]. Nous cherchons à combler ce fossé sous l'hypothèse qu'une représentation sémantique plus élevée que les fréquences des mots visuels permet d'améliorer la qualité de recherche.

Les résultats de l'AFC sur l'analyse de données textuelles montre que les axes factoriels de l'AFC (déterminés par des groupes de mots, appelés les *métaclés*) correspondent à des concepts plus généraux que les mots individuels et appréhendent les *thèmes* dans un corpus [Mor04, KBC06]. L'application de l'AFC sur un tableau de contingence qui croise des images et des mots visuels permet de trouver des thèmes au niveau sémantique plus élevés que les mots visuels individuels. L'utilisation de ces thèmes plutôt que celle des mots visuels améliorera la qualité de recherche. À titre d'exemple, nous montrons dans la figure 3.4, la projection de 4 catégories de la base Caltech-4 (cf. Section 3.6) sur les axes factoriels. Il est clair que les catégories sont bien distinguées avec 2 axes seulement.

Soit $\mathbf{X}$ le tableau de contingence croisant les M images et les N mots visuels (supposons que $N < M$). Nous faisons l'AFC de ce tableau et obtenons les axes factoriels correspondant aux thèmes. Après avoir écarté la première valeur propre (c-à-d. la valeur propre triviale égale à 1), nous ne conservons que les $K(K < N)$ premières plus grandes valeurs propres et les vecteurs propres associés. Ces K vecteurs propres constituent une base orthonormée de l'espace réduit (appelé aussi l'espace des facteurs ou l'espace

5. Notons cependant que les mots visuels sont déjà au niveau plus élevé que les pixels dans la description sémantique du contenu du images.

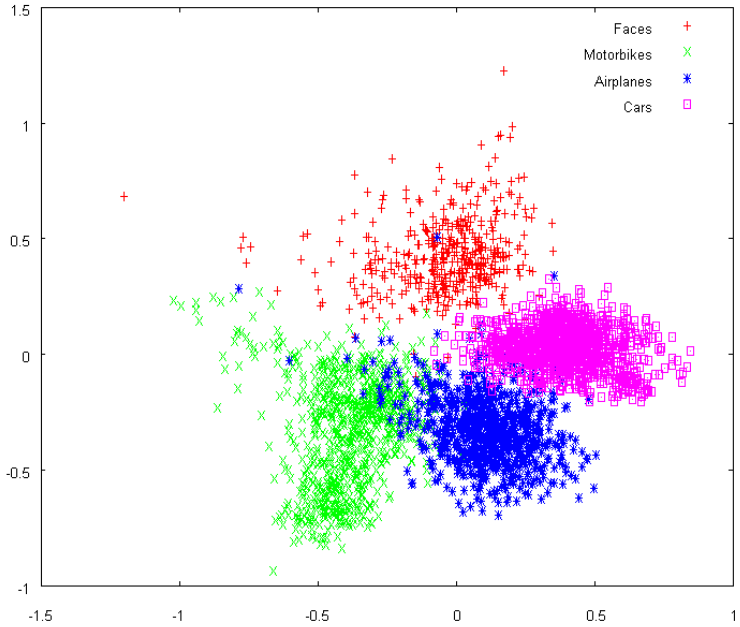


FIGURE 3.4 – Projection de 4 catégories (*Faces*, *Motorbikes*, *Airplanes*, et *Cars*) de la base Caltech-4 sur les axes factoriels de l'AFC.

factoriel). La projection des images sur ces axes fournit une représentation réduite des images. Le nombre de dimensions du problème passe de N à K. L'intérêt de l'AFC est donc de combler le *fossé sémantique* dans la description du contenu d'images et de simultanément de réduire le nombre de dimensions.

La nouvelle représentation des images sur les axes factorielle est obtenue par la formule 3.23. Les mesures de similarité/dissimilarité s'appliquent à cette nouvelle représentation. Par exemple, la mesure de similarité du cosinus entre l'image i et l'image i' dans la base s'obtient par :

$$\cos(i,i') = \frac{\sum_{\alpha=1}^{K} z_{i\alpha} z_{i'\alpha}}{\sqrt{\sum_{\alpha'=1}^{K} z_{i\alpha'}^2} \sqrt{\sum_{\alpha''=1}^{K} z_{i'\alpha''}^2}} \tag{3.55}$$

où $z_{i\alpha}$ est la coordonnée de l'image i sur l'axe α et $z_{i'\alpha}$ est la coordonnée de l'image i' sur l'axe α.

Le choix de K

L'AFC étant une méthode de réduction de dimension, le choix du nombres d'axes à conserver, K, peut être effectué en analysant la contribution des axes à l'inertie totale (qui est proportionnelle à la valeur propre associée aux axes, λ_α). La méthode

décrite dans [LMP97] (cf. Section 3.3. Analyse factorielle discriminante, pages 251–283) peut servir pour déterminer K. En effet, les premières valeurs propres décroissent souvent très rapidement mais ensuite la décroissance devient beaucoup plus lente. La figure 3.5 montre les valeurs propres de l'AFC sur la base Caltech-4 (cf. section 3.6). La décroissance des valeurs propres est assez typique des tableaux creux. Les premières valeurs propres décroissent rapidement mais après l'axe 50, la décroissance est très lente, presque linéaire. Dans ce cas, nous pouvons prendre 50 premiers axes.

En pratique, K est choisi empiriquement et dépend du but de l'analyse. Dans le cas de découverte de thèmes dans un corpus de textes [Mor04, KBC06], on utilise souvent 30 premiers axes pour l'interprétation des résultats car la décroissance des valeurs propres devient souvent lente après l'axe 30. Pour trouver d'autres axes (correspondant aux thèmes « cachés »), on enlève les documents et les mots ayant une forte contribution et on refait une deuxième AFC. Les documents et les mots enlevés sont re-projetés sur les nouveaux axes comme des éléments supplémentaires [6]. Les premiers axes (dont la direction est déterminés par les éléments ayant une forte contribution) empêchent de trouver d'autres axes qui ne sont pas perpendiculaires avec les premiers axes mais ayant une inertie élevée.

Pour la tâche de recherche d'images, le choix d'une valeur optimale de K dépend fortement de ce que nous voulons chercher. En effet, une faible valeur de K diminue l'effet du bruit, réduit la variance intra-classe et permet de trouver plutôt les catégories que les images. Par exemple, si la requête est une voiture, les voitures avec différentes formes ou différents marques pourront être retournées. Par contre, une grande valeur de K augmente la discrimination entre les images et permet ainsi de trouver des images presque identiques à l'image requête (par ex. détection de copies).

3.5 Accélération de la recherche d'images par l'AFC

L'application de l'AFC sur les images permet à la fois de réduire le nombre de dimensions de N à K et d'améliorer la qualité de recherche (la *précision*) par rapport à une pondération *tf*idf*. Dans le cas où K est petit, les techniques d'indexation présentées dans le chapitre 2 peuvent être utilisées pour accélérer la recherche. Cependant, le nombre d'axes conservés K est généralement assez élevé (par ex. 100). Et la représentation réduite des images (issue de l'AFC) n'est plus creuse. L'utilisation directe des techniques de fichiers inversés devient impossible. Nous cherchons donc une autre façon d'accélérer la recherche d'images sans réduire la qualité des résultats. Notre approche se base sur deux indicateurs de l'AFC qui aident à interpréter les résultats. Nous présentons ensuite les deux indicateurs de l'AFC et deux systèmes de fichiers inversés construits à partir des indicateurs correspondant utilisés pour accélérer la recherche.

6. Puisque les axes factoriels doivent être perpendiculaires, la direction d'un axe dépendra des axes précédents. Si on enlève les éléments ayant une forte contribution et si on les traite comme des éléments supplémentaires on peut trouver d'autres thèmes « cachés ».

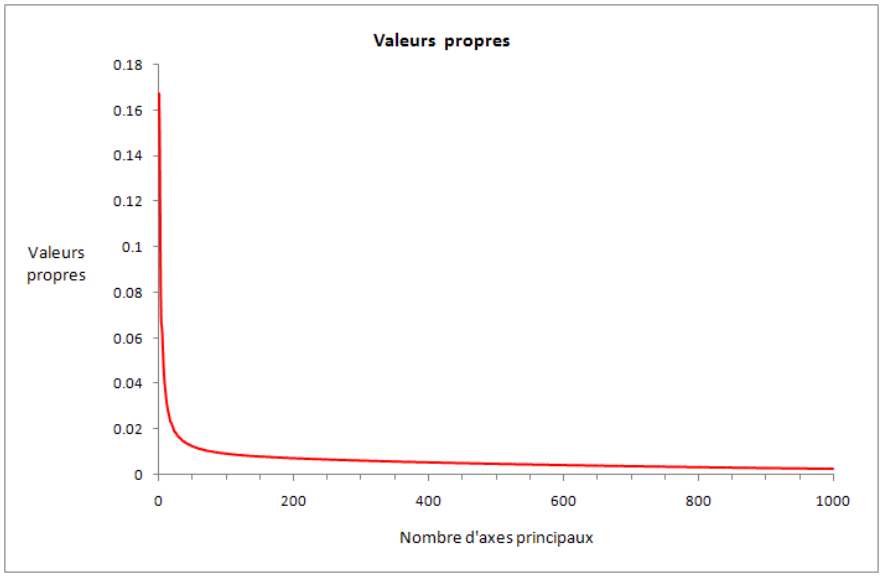


FIGURE 3.5 – La séquence des valeurs propres de l'AFC sur la base Caltech-4.

3.5.1 Les indicateurs de l'AFC

L'AFC nous fournit deux indicateurs pertinents pour interpréter les résultats : la *contribution* d'un point (image ou mot visuel) à un axe et la *qualité de représentation* d'un point sur un axe.

Définition 3.1 (Contribution) – *La contribution d'un point-ligne i (image i) à l'axe α est définie par :*

$$\mathrm{ctr}_\alpha(i) = f_{i.}\frac{z_{i\alpha}^2}{\lambda_\alpha} \tag{3.56}$$

$$\forall\alpha, \sum_{i=1}^{M}\mathrm{ctr}_\alpha(i) = 1 \tag{3.57}$$

De la même façon, la contribution d'un point-colonne j (mot visuel j) se définit par :

$$\mathrm{ctr\_mot}_\alpha(j) = f_{.j}\frac{w_{j\alpha}^2}{\lambda_\alpha} \tag{3.58}$$

$$\forall\alpha, \sum_{j=1}^{N}\mathrm{ctr\_mot}_\alpha(j) = 1 \tag{3.59}$$

La contribution d'un point i à l'axe α permet de savoir dans quelle proportion un point i contribue à l'inertie λ_α du nuage projeté sur l'axe α. Les points ayant une forte contribution à un axe ont un effet sur la direction de l'axe.

Définition 3.2 (Qualité de représentation) – *La qualité de représentation (appelée aussi le cosinus carré) d'un point-ligne i (image i) sur l'axe α est définie par :*

$$\cos^2_\alpha(i) \quad = \quad \frac{z^2_{i\alpha}}{\sum_{\alpha'} z^2_{i\alpha'}} \tag{3.60}$$

Notons que pour tout i :

$$\sum_\alpha \cos^2_\alpha(i) \quad = \quad 1$$

De la même façon la qualité de représentation d'un point-colonne j (mot visuel j) se définit par :

$$\cos\_\,\mathrm{mot}^2_\alpha(j) \quad = \quad \frac{w^2_{j\alpha}}{\sum_{\alpha'} w^2_{j\alpha'}} \tag{3.61}$$

et pour tout j :

$$\sum_\alpha \cos\_\,\mathrm{mot}^2_\alpha(j) \quad = \quad 1$$

Le cosinus carré permet d'apprécier si un point est bien représenté sur un axe factoriel (un *facteur*). On l'appelle aussi la *contribution relative* du facteur à la position du point en examen.

Pour analyser la proximité entre points, on s'intéresse surtout aux points ayant un cosinus carré élevé. Les proximités entre ces points, observées dans le sous-espace factoriel, donnent une bonne image de leurs proximités réelles.

3.5.2 Fichiers inversés

Dans [SZ03b, NS06], la représentation creuse des images (modèle de sac-de-mots-visuels) a été exploitée pour accélérer la recherche via un système de fichiers inversés (cf. Section 2.3.3). Il s'agit d'une technique empruntée à la recherche d'information dans les textes utilisant le modèle de sac-de-mots et faisant l'hypothèse d'indépendance des mots textuels, un mot apparaissant indépendamment des autres. Mais dans le cas des images il existe une certaine dépendance entre les mots visuels. Plusieurs mots visuels (au lieu d'un seul mot visuel) étant nécessaires pour décrire un objet (cf. Section 3.4.3). L'hypothèse d'indépendance des mots est encore moins réaliste que dans le contexte de recherche de textes. De plus, dans le cas de la recherche de textes, le nombre de mots-clés de la recherche est peu élevé, alors que dans le cas de la recherche d'images, le nombre de mots-clés visuels est beaucoup plus important. Les fichiers inversés basés sur les mots visuels seront donc moins efficaces.

Notre système de fichiers inversés n'est pas construit directement à partir des mots visuels mais se base sur les thèmes issus des axes de l'AFC. Un thème correspond à une partie (positive ou négative) d'un axe de l'AFC et il est expliqué par un groupe de mots

visuels (ou un groupe d'images) ayant une forte contribution à cet axe. L'hypothèse d'indépendance des thèmes est plus réaliste et le système ne souffre donc pas de la longueur des requêtes. Nous cherchons à associer les images aux thèmes auxquels les images appartiennent. Un fichier inversé est construit pour chaque thème et contient des images appartenant à ce thème.

Étant donnée une requête, la recherche consiste d'abord à déterminer les thèmes auxquels la requête appartient. Nous prenons ensuite les fichiers inversés associés, les fusionnons et nous filtrons les images qui sont non pertinentes (les images qui partagent moins de thèmes avec la requête) pour former une *liste d'images candidates*. La recherche proprement dite se fait dans la *liste d'images candidates*. Il s'agit d'une recherche séquentielle dans une liste de petite taille.

Nous présentons ci-dessous deux systèmes de fichiers inversés basés sur deux indicateurs pertinents de l'AFC. Puisque les axes factoriels de l'AFC correspondent aux thèmes, nous construisons, pour un axe, deux fichiers inversés correspondant aux deux thèmes (un pour la partie négative et l'autre pour la partie positive). Le nombre total de fichiers inversés est de $2K$. Nous distinguons deux systèmes de fichiers inversés selon les informations utilisées pour déterminer les thèmes.

Définition 3.3 (Fichier inversé basé sur la contribution) – *Étant donné un seuil $\varepsilon > 0$, les deux fichiers inversés basés sur la contribution associés à l'axe α, notés CF_α^+ (pour la partie positive) et CF_α^- (pour la partie négative) sont définis par :*

$$CF_\alpha^+ = \{i \mid \mathrm{ctr}_\alpha(i) > \varepsilon \text{ et } z_{i\alpha} > 0\} \quad (3.62)$$

$$CF_\alpha^- = \{i \mid \mathrm{ctr}_\alpha(i) > \varepsilon \text{ et } z_{i\alpha} < 0\} \quad (3.63)$$

où $z_{i\alpha}$ est la coordonnée de l'image i sur l'axe α.

Le seuil ε est choisi égal à la contribution moyenne. Puisque la somme des contributions est égale à 1, le seuil ε choisi est donc égal à $\frac{1}{M}$ où M est le nombre d'images dans la base.

Définition 3.4 (Fichier inversé basé sur la qualité de représentation) – *Étant donné un seuil $\varepsilon > 0$, les deux fichiers inversés basés sur la qualité de représentation associés à l'axe α, notés QF_α^+ (pour la partie positive) et QF_α^- (pour la partie négative) sont définis par :*

$$QF_\alpha^+ = \{i \mid \cos_\alpha^2(i) > \varepsilon \text{ et } z_{i\alpha} > 0\} \quad (3.64)$$

$$QF_\alpha^- = \{i \mid \cos_\alpha^2(i) > \varepsilon \text{ et } z_{i\alpha} < 0\} \quad (3.65)$$

où $z_{i\alpha}$ est la coordonnée de l'image i sur l'axe α.

Comme les fichiers inversés basés sur la contribution, le seuil ε pour les fichiers inversés basés sur la qualité de représentation peut être choisi égal à la qualité de la représentation moyenne. Puisque la somme des cosinus carrés sur tous les axes est égale à 1, le seuil ε est donc déterminé par $\frac{1}{K}$ avec K est le nombre d'axes conservés.

3.5.3 Algorithme de recherche

L'association des images aux thèmes obtenus par l'AFC forme une représentation creuse des images [7]. Ceci permet d'utiliser la technique des fichiers inversés. Notons que le nombre de thèmes est beaucoup plus petit que le nombre de mots visuels et qu'une image n'appartient qu'à un petit nombre de thèmes. Donc, la représentation creuse basée sur les thèmes est plus compacte que celle basée directement sur les mots visuels. Cependant les mesures de similarités/dissimilarités ne peuvent pas se calculer directement à partir de la représentation basée sur les thèmes à cause de la perte d'information (la discrimination des images) de la nouvelle représentation creuse. Nous proposons donc un algorithme de recherche en deux étapes qui à la fois exploite la représentation creuse (la représentation basée sur les thèmes) pour accélérer la recherche et prend en compte le pouvoir discriminant de la représentation dense des images (la représentation par les coordonnées factorielles) pour améliorer la qualité des résultats.

La première étape de l'algorithme consiste à filtrer les images non pertinentes pour la requête et conduit à une liste d'images candidates en utilisant des fichiers inversés. La seconde étape raffine la recherche par balayage séquentiel dans la liste d'images candidates. La similarité des images sera mesurée à partir de la représentation dense. L'algorithme 1 décrit les étapes principales de la procédure de recherche.

Algorithme 1 : Algorithme de recherche basé sur les fichiers inversés

1. Projeter la requête $\mathbf{r}$ dans l'espace réduit par la formule de transition 3.42
2. Déterminer les thèmes auxquels la requête appartient et prendre les fichiers inversés associés
3. Fusionner les fichiers inversés et calculer la fréquence des images
4. Filtrer les images non pertinentes pour construire la liste d'images candidates
5. Chercher les plus proches voisins de la requête $\mathbf{r}$ dans la liste d'images candidates

Détermination des thèmes de la requête

Les thèmes associés à une requête sont déterminés de la même manière que la construction des fichiers inversés, en prenant les axes dont la contribution (ou la qualité de représentation) est supérieure au seuil ε. Lorsqu'un axe est choisi, le thème correspondant à la partie négative ou positive sera associé à la requête si sa projection sur l'axe se trouve dans la partie respective (négative ou positive). Comme l'image requête ne contribue pas à la construction des axes (elle possède une contribution nulle), sa *pseudo contribution*, définie ci-dessous, est utilisée à la place de la contribution dans le cas où les fichiers inversés basés sur la contribution sont utilisés.

7. Nous distinguons ici la représentation dense et la représentation creuse des images basée sur les thèmes de l'AFC. La première est la projection des images sur les axes factoriels tandis que la seconde est une version binaire de la première (via l'association des images aux thèmes) et ne contient que des 0 (l'image n'appartient pas au thème correspondant) et des 1 (l'image appartient au thème correspondant).

Définition 3.5 (Pseudo contribution) – *Soit* $\mathbf{r}$ *une image qui ne participe pas à la construction des axes (par ex., l'image requête) représentée par la distribution des mots visuels* $[r_1\ r_2\ \ldots\ r_N]$. *La pseudo contribution de* $\mathbf{r}$ *à l'axe* α *est définie par :*

$$\begin{aligned} \mathrm{ctr}_\alpha(\mathbf{r}) &= \mathrm{m}(\mathbf{r})\frac{\left(z_\alpha^{(\mathbf{r})}\right)^2}{\lambda_\alpha} \\ \text{avec } \mathrm{m}(\mathbf{r}) &= \frac{\sum_{i=1}^{N} r_i}{x_{..}} \end{aligned} \tag{3.66}$$

où $\mathrm{m}(\mathbf{r})$ *est la pseudo masse de l'image* $\mathbf{r}$ *et* $z_\alpha^{(\mathbf{r})}$ *est la projection de* $\mathbf{r}$ *sur l'axe* α.

Filtrage des images non pertinentes

Soit $\mathbb{F} = \left\{F_{\alpha_1}^{\sigma_1}, F_{\alpha_2}^{\sigma_2}, \ldots, F_{\alpha_n}^{\sigma_n}\right\}$ où $\alpha_i \in [1;K], \sigma_i \in \{+,-\}$ l'ensemble des n fichiers inversés associés à une requête $\mathbf{r}$ obtenus par l'étape de détermination des thèmes. Nous donnons d'abord quelques définitions avant de présenter le processus de filtrage des images non pertinentes.

Définition 3.6 (Fréquence des images) – *La fréquence d'une image* i, *notée* $\mathrm{freq}(i)$, *est définie comme le nombre de fichiers inversés auxquels l'image* i *appartient.*

$$\mathrm{freq}(i) = \left|\left\{F_\alpha^\sigma \mid i \in F_\alpha^\sigma\right\}\right| \tag{3.67}$$

où $F_\alpha^\sigma \in \mathbb{F}$ *est un fichier inversé associé à la requête et* $|.|$ *désigne la cardinalité d'un ensemble.*

Définition 3.7 (Liste fusionnée) – *Nous appelons la liste fusionnée, notée* $\mathbb{L}$, *le résultat de la fusion des fichiers inversés dans* $\mathbb{F}$ *:*

$$\mathbb{L} = \{\langle i, \mathrm{freq}(i)\rangle\} \tag{3.68}$$

où

$$i \in \bigcup_{F_\alpha^\sigma \in \mathbb{F}} F_\alpha^\sigma$$

et $\mathrm{freq}(i)$ *est la fréquence de l'image* i.

La fréquence d'une image signifie également le nombre de thèmes que l'image partage avec la requête. Les images qui ne partagent aucun thème avec la requête ne se trouvent pas dans la liste fusionnée. Après avoir fusionné les fichiers inversés, nous obtenons une liste d'images avec leur fréquence. Le filtrage des images non pertinentes pour la requête se fait sous l'hypothèse de pertinence « *plus la fréquence d'une image est élevée, plus l'image est pertinente pour la requête* ».

Soit n le nombre de thèmes d'une requête $\mathbf{r}$, soit $\mathcal{H}_\theta = \{i \mid \mathrm{freq}(i) \geq \theta\}, \theta = 1..n$ l'ensemble d'images dont la fréquence est supérieure ou égale à θ. Nous avons :

$$\mathcal{H}_1 \supseteq \mathcal{H}_2 \supseteq \cdots \supseteq \mathcal{H}_n$$

Le filtrage d'images non pertinentes consiste à déterminer un seuil $\hat{\theta}$ tel que $\mathcal{H}_{\hat{\theta}}$ (nous appelons $\mathcal{H}_{\hat{\theta}}$ la *liste d'images candidates*) contient des images pertinentes pour la requête $\mathbf{r}$. Pour une raison d'efficacité, $\hat{\theta}$ doit être le plus grand possible car $|\mathcal{H}_\theta|$ décroît lorsque θ augmente. Une grande valeur de $\hat{\theta}$ filtre beaucoup d'images et peut conduire à une liste d'images candidates vide. Par contre, un petit $\hat{\theta}$ conserve un nombre important d'images et dégrade l'efficacité de l'étape de raffinement. Une solution naïve est de choisir le seuil $\hat{\theta}$ égal à $\lceil \frac{n}{2} \rceil$. Cependant, avec un grand n, le seuil $\lceil \frac{n}{2} \rceil$ sera trop contraignant et la liste d'images candidates deviendra vide ou contiendra moins de k images (il s'agit d'une recherche de k plus proches voisins). Pour garantir que le système retourne toujours k images, il faut que la liste d'images candidates contienne au moins k images. Partant de cette idée, nous proposons de choisir le seuil $\hat{\theta}$ tel que la liste d'images candidates $\mathcal{H}_{\hat{\theta}}$ contient au moins s images (par ex. 500). La valeur de s peut être déterminée en fonction de k (par ex. $3k$) et/ou de la taille de la base (par ex. 5% de la base).

Recherche interactive

En faisant l'hypothèse de pertinence (la pertinence d'une image dans $\mathcal{H}_\theta$ est plus élevée que celle d'une image dans $\mathcal{H}_{\theta-1}$), nous proposons une procédure de recherche avec interaction de l'utilisateur. Le système présente successivement les images par pertinence décroissante. Le système cherche d'abord un seuil $\hat{\theta}$ tel que $\mathcal{H}_{\hat{\theta}}$ contient un nombre assez petit d'images (par ex. 100) et présente ces images à l'utilisateur. S'il veut plus d'images, les images dans $\mathcal{H}_{\hat{\theta}-1} \backslash \mathcal{H}_{\hat{\theta}}$ seront présentées. Ensuite, les images dans $\mathcal{H}_{\hat{\theta}-2} \backslash \mathcal{H}_{\hat{\theta}-1}$ seront retournées si l'utilisateur en veut plus et ainsi de suite. La recherche s'arrête lorsque toutes les images dans $\mathcal{H}_1$ sont examinées ou que l'utilisateur arrête la recherche. La procédure de recherche est décrite dans l'algorithme 2.

Algorithme 2 : Algorithme de recherche interactive

```
1  Choisir un seuil θ̂ tel que H_θ̂ contient un certain nombre d'images (par ex. 100)
   et présenter les images de H_θ̂ à l'utilisateur.
   θ = θ̂
   tant que θ > 1 faire
2      si l'utilisateur veut encore plus d'images alors
           présenter les images de H_{θ-1} \ H_θ à l'utilisateur
           θ = (θ − 1)
3      sinon
           arrêter la recherche
       fin
   fin
```

Pendant l'interaction, les informations comme le nombre de thèmes associés à la requête $\mathbf{r}$, n et la cardinalité des $\mathcal{H}_\theta$, $|\mathcal{H}_\theta|$ seront éventuellement nécessaires pour aider l'utilisateur à arrêter ou continuer la recherche.

3.6 Expérimentations

3.6.1 Bases d'images

Nous avons utilisé plusieurs bases d'images connues en recherche et catégorisation d'images.

Caltech-4 Cette base est utilisée dans [SRE+05] pour la tâches de catégorisation d'images. La base contient 4 090 images réparties en catégories : *faces* (435 images), *motorbikes* (800 images), *airplanes* (800 images), *backgrounds* (900 images) et *car-rears* (1 155 images). La figure 3.6 montre certaines images extraites de la base.

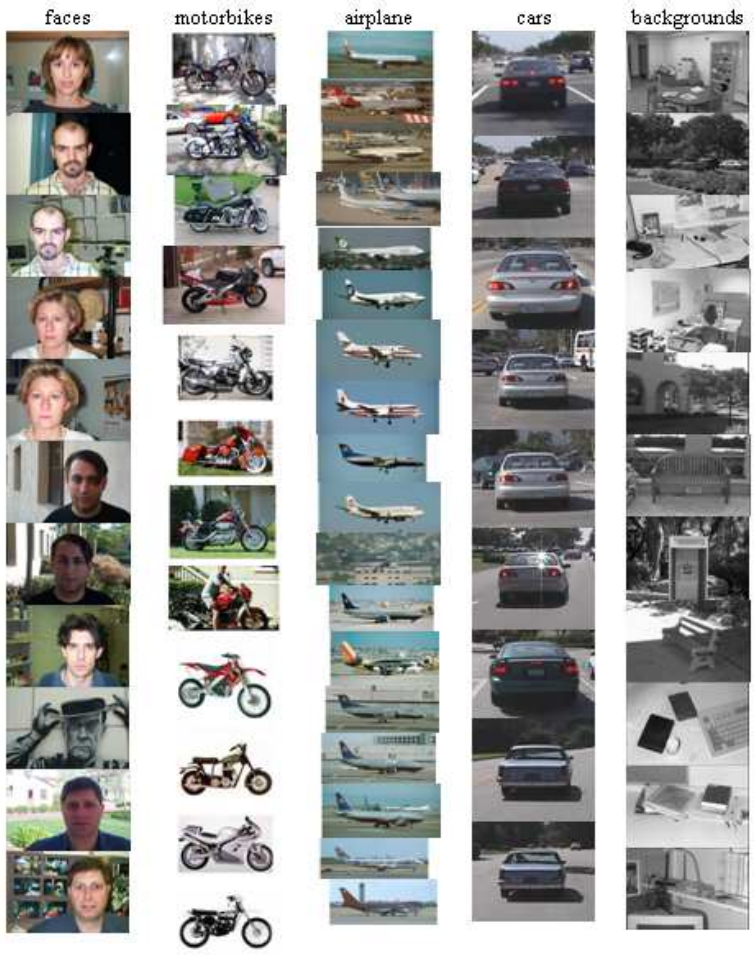


FIGURE 3.6 – Images extraites de la base Caltech-4

Caltech-101 [FFFP04] Cette base contient 9 144 images réparties en 102 catégories. Nous avons enlevé la catégorie *Faces_easy* (435 images) car cette catégorie n'est autre que la version rognée (les images sont rognées et ne contiennent que le visage) de la

catégorie *Faces* (435 images). Il reste donc 8 709 images (101 catégories).

Nistér-Stewénius [NS06] La base contient 2 250 groupes de 4 images prises sur une même scène avec différentes positions de l'appareil photo. Le nombre total d'images de la base est de 10 200. Quelques images de la base sont montrées dans la figure 3.7.

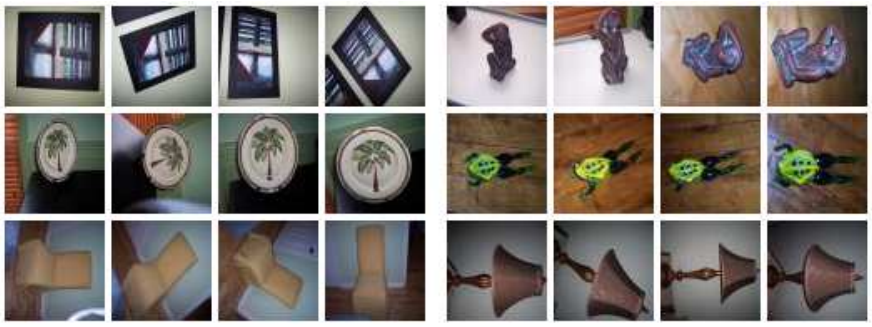


FIGURE 3.7 – Images extraites de la base Nistér-Stewénius

3.6.2 Construction des mots visuels

Les données de la base Caltech-4 sont fournies par les auteurs de [SRE+05]. Les mots visuels sont construits à partir d'environ 300 000 descripteurs extraits aléatoirement. La taille du vocabulaire visuel est de 2 224. Nous obtenons ainsi un tableau de contingence à 4 090 lignes et à 2 224 colonnes.

Pour les autres bases, les mots visuels sont obtenus selon le processus décrit dans la section 3.4.1. Le nombre de mots visuels est calé à 5 000.

1. *Détection des points d'intérêt* – Nous avons utilisé le détecteur *Hessien-affine* qui a prouvé être un des meilleurs détecteurs [MTS+05].
2. *Caractérisation des points d'intérêt* – Le descripteur SIFT [Low04] a été utilisé pour décrire la région locale autour des points d'intérêt. Les descripteurs SIFT donnent les meilleurs résultats dans la mise en correspondances des images (image matching) [MS05].
   Les deux étapes sont effectuées par le logiciel *extract_feature* de Mikolajczyk et Schmid [MS04] avec les options *-haraff -sift -thres 100*).
3. *Clustering des descripteurs locaux* – Nous avons réimplanté l'algorithme standard LLoyd de k-means en C++ sur GPUs utilisant CUBLAS et CUDA [Was06] pour accélérer la quantification. Avec 1 625 980 descripteurs locaux SIFT à 128 dimensions et $k = 5\,000$ *clusters*, le temps d'exécution (30 itérations) sur une machine Intel Xeon bi-processeurs Quad-core, 3.2GHz, 8GO de mémoire vive, munie d'une carte graphique NIVIDA GeForce GTX 280 (1GO de mémoire) est d'environ 7.5 minutes (soit 15 secondes/itération).

TABLE 3.2 – Tableaux de contingence pour les 3 bases d'images expérimentées. M est le nombre d'images et N est le nombre de mots visuels.

Bases ↓	M	N	Taux du remplissage	Temps de calcul (s)	
				SIFT	Affectation
Caltech-4	4 090	2 224	0.130	-	-
Caltech-101	8 709	5 000	0.095	399.5 (0.0489)	160.0 (0.0184)
Nistér-Stewénius	10 200	5 000	0.218	1 201.1 (0.1178)	419.5 (0.0392)

TABLE 3.3 – Schémas de pondération *tf*idf* expérimentés.

Méthodes	*idf*	Note
*tf*idf*	$\log\left(\frac{M}{m_j}\right)$	*idf* standard
*tf*idf*-1	$\max\left(0, \log\left(\frac{M-m_j}{m_j}\right)\right)$	*idf* probabiliste
*tf*idf*-2	1	Sans pondération

3.6.3 Tableaux de contingence

Le tableau 3.2 décrit le taux effectif de remplissage des tableaux de contingence correspondant à 3 bases d'images utilisées. Nous montrons également dans ce tableau le temps de calcul nécessaire pour les obtenir. La colonne "SIFT" présente le temps (en seconde) pour calculer les descripteurs SIFT et le temps pour affecter les descripteurs aux mots visuels correspondants est montré dans la colonne "Affectation". L'affectation des descripteurs est effectuée sur 2 GPU. Les chiffres dans les parenthèses montrent les temps de calcul moyens pour une image.

3.6.4 Méthodes de référence

La pondération *tf*idf* (qui est utilisée dans *Video google* [SZ03b]) est considérée comme une méthode de référence. Chaque élément x_{ij} (ligne i, colonne j) du tableau de contingence $\mathbf{X}$ est normalisé à tf_{ij}

$$tf_{ij} = \frac{x_{ij}}{x_{i.}}$$

et est pondéré par idf_j

$$idf_j = \log\left(\frac{M}{m_j}\right)$$

où M est le nombre d'images de la base et m_j est le nombre d'images qui contient le mot visuel j. Nous avons également expérimenté d'autres schémas de pondération. Le tableau 3.3 résume les schémas utilisés.

La méthode LSA est effectuée sur le schéma de pondération qui donne le meilleur MAP. Nous avons comparé l'AFC avec une des méthodes de modélisation probabiliste, le PLSA [Hof99a] [8]. Dans ce cas, les images sont représentées par les distributions des thèmes par images : les probabilités $P(thème \mid image)$. Puisque le PLSA utilise l'algorithme EM pour estimer les paramètres (comme la plupart des méthodes de modélisation probabiliste), le résultat obtenu n'est qu'une solution locale et dépend de la configuration initiale. Pour chaque expérimentation, nous avons lancé le PLSA 10 fois avec différentes configurations initiales et le nombre d'itérations pour l'algorithme EM est fixé à 100. Les mesures d'évaluation sont calculées à chaque fois et les moyennes sont récupérées.

3.6.5 Mesures de similarité/dissimilarité

Nous utilisons la similarité du cosinus comme une mesure de similarité pour les images comme la plupart des systèmes de recherche d'information. La distance euclidienne (L_2) et la distance de Manhattan (L_1) sont expérimentées à titre de comparaison. Nous avons aussi expérimenté la divergence symétrique de Jensen-Shannon (notée mesure J-S) pour le PLSA. Cette mesure J-S se base sur la divergence de Kullback Leibler. La mesure J-S de deux distributions $\mathbf{x}$ et $\mathbf{y}$ est obtenue par :

$$d_{\mathrm{JS}}(\mathbf{x},\mathbf{y}) \quad = \quad \frac{1}{2}\left(d_{\mathrm{KL}}(\mathbf{x},\frac{\mathbf{x}+\mathbf{y}}{2}) + d_{\mathrm{KL}}(\mathbf{y},\frac{\mathbf{x}+\mathbf{y}}{2})\right) \tag{3.69}$$

où d_{KL} est la divergence de Kullback Leibler :

$$d_{\mathrm{KL}}(\mathbf{x},\mathbf{y}) \quad = \quad \sum_i x_i \log \frac{x_i}{y_i} \tag{3.70}$$

3.6.6 Mesures d'évaluation

Nous utilisons les mesures de performances standard des systèmes de recherche d'information décrits dans la section 1.6 du chapitre 1 pour évaluer la performance des méthodes de recherches : la précision, le rappel, la précision moyenne (MAP) et le $MANR$ (Mean average normalized rank).

Nous calculons la précision aux 10, 20, 50, 100 et 200 premières images retournées pour les bases Caltech-4 et Caltech-101. Dans le cas de la base de Nistér-Stewénius, nous calculons la précision aux 3 premières images puisqu'il n'y a que 3 images pertinentes pour une requête. Le score N-S utilisé dans [NS06] a été aussi calculé. Ce score est le nombre d'images pertinentes (y compris la requête) parmi les 4 premières images retournées.

Puisque toutes les images de la base doivent être examinées dans une recherche exhaustive, le *rappel*, dans ce cas, ne porte pas d'information (le *rappel* est toujours égal à 1). Par contre, nous pouvons calculer le *rappel* aux k premières images retournées. Mais, avec un k donné, le *rappel* est proportionnel à la *précision*.

8. Nous avons utilisé l'implémentation du PLSA en Matlab de J. Verbeek qui se trouve à `http://lear.inrialpes.fr/~verbeek/software.php` pour les expérimentations.

Pour évaluer la capacité de rangement des méthodes, nous calculons la précision à chaque image pertinente pour la requête et obtenons la *courbe de précision–rappel* et la précision moyenne (MAP). Nous utilisons aussi la mesure $MANR$.

3.6.7 Tests statistiques

Nous avons utilisé le test non paramétrique classique : le test de Wilcoxon. Le seuil pour p-valeur est fixé à 0.05. Dans les tableaux de résultats, nous montrons les chiffres :

- **en gras et soulignés** : le(s) meilleur(s) résultat(s) des méthodes utilisées sur une mesure d'évaluation
- **en gras seulement** : les résultats dont la différence du meilleur (du tableau) n'est pas statistiquement significative.

3.6.8 Recherche exhaustive

Les expérimentations présentées dans cette section ont été effectuées par une recherche exhaustive (toutes les images dans la base doivent être comparées avec la requête) pour le PLSA et l'AFC. Cependant, la technique des fichiers inversés peut être appliquée pour la méthode *tf*idf* grâce à sa représentation creuse.

AFC versus d'autres méthodes

Nous comparons la performance de l'AFC et celle des méthodes de références : *tf*idf*, LSA et PLSA. La similarité du cosinus est utilisée pour mesurer la similarité entre les images de la base et la requête.

La figure 3.8 présente les courbes de *précision–rappel* des différentes méthodes : *tf*idf*, LSA, PLSA et AFC calculées pour les bases Caltech-4, Caltech-101 et Nistér-Stewénius respectivement. Le nombre d'axes, K (correspondant au nombre de thèmes pour la méthode PLSA et au nombre d'axes conservés pour la méthode AFC) est choisi empiriquement de façon à obtenir le meilleur MAP pour le PLSA. Pour la base Caltech-4, le meilleur MAP s'obtient avec $K = 7$. Ce résultat est cohérent avec les résultats obtenus dans [SRE+05].

Les tableaux 3.4 et 3.5 présentent les résultats expérimentaux sur les deux bases d'images : Caltech-4 et Caltech-101 avec différentes mesures d'évaluation : la précision des méthodes aux 10, 20, 50, 100 et 200 premières images retournées (cf. les colonnes P@10, P@20, P@50, P@100 et P@200), la précision moyenne (MAP) et le $MANR$. Pour la base Nistér-Stewénius, nous fournissons le score N-S, la précision aux 3 premières images retournées, la précision moyenne et la mesure $MANR$ dans le tableau 3.6. Nous avons testé le PLSA avec la similarité du cosinus (PLSA-1) et la mesure J-S (PLSA-2).

Les résultats montrent que l'AFC, le LSA et le PLSA font beaucoup mieux qu'une pondération *tf*idf* simple. Ceci vérifie l'hypothèse que nous avons posée dans la section 3.4.3 « *une représentation sémantique plus élevée que les fréquences des mots visuels permet d'améliorer la qualité de recherche* ». En comparaison avec le LSA et le PLSA, l'AFC donne toujours de meilleurs résultats quel que soit le nombre d'axes conservés K. Lorsque le nombre de catégories de la base augmente (ou que les images ne sont pas

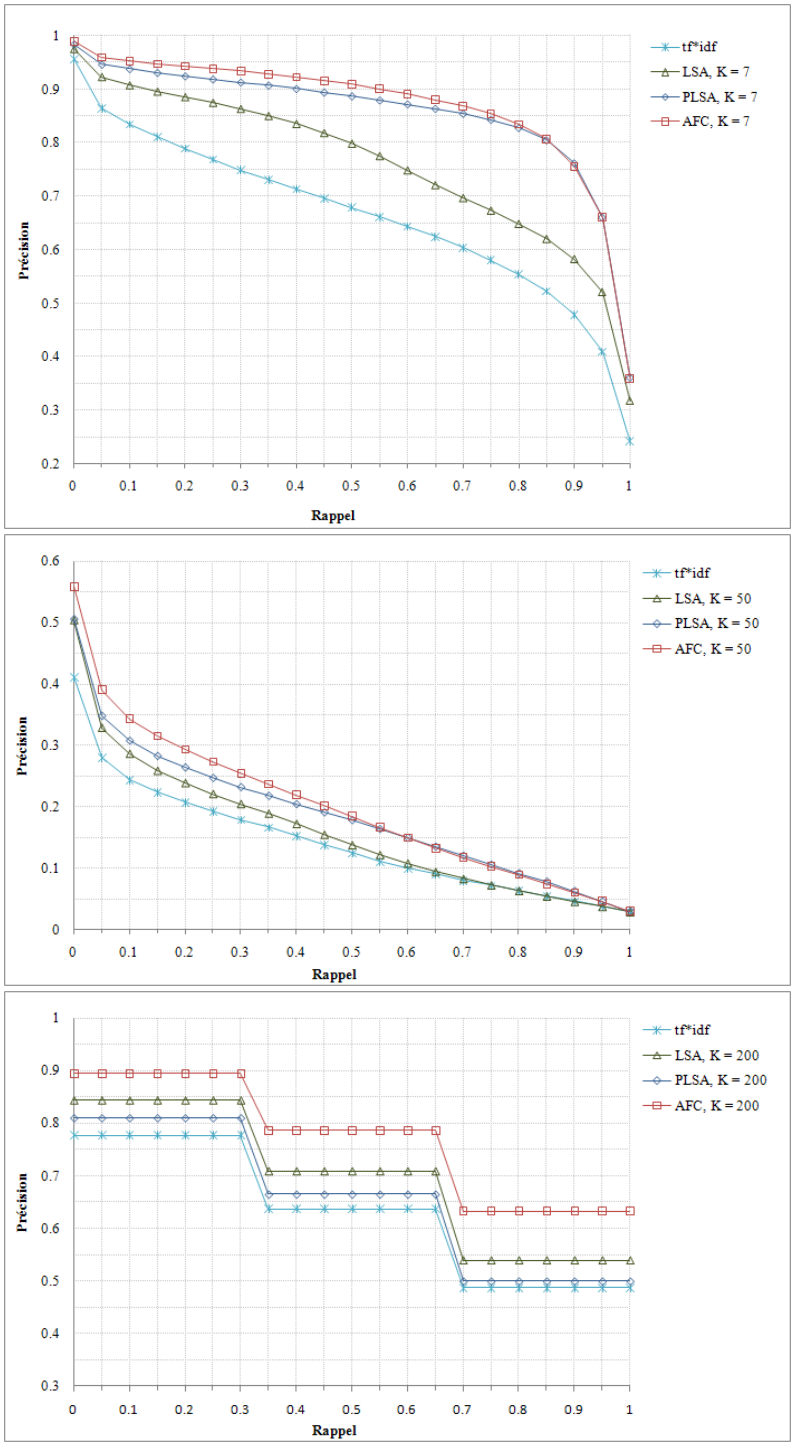


FIGURE 3.8 – Courbes de *précision–rappel* des différentes méthodes avec la similarité de cosinus pour les bases Caltech-4 (en haut), Caltech-101 (au milieu) et Nistér-Stewénius (en bas).

TABLE 3.4 – Résultats des différentes méthodes sur la base Caltech-4.

Méthodes ↓	P@10	P@20	P@50	P@100	P@200	MAP	$MANR$
*tf*idf*	0.879	0.868	0.849	0.828	0.795	0.663	0.128
*tf*idf*-1	0.883	0.872	0.853	0.833	0.800	0.670	0.127
*tf*idf*-2	0.870	0.857	0.837	0.814	0.778	0.641	0.133
LSA, $K = 5$	0.879	0.876	0.873	0.867	0.853	0.754	0.089
LSA, $K = 7$	0.918	0.915	0.907	0.896	0.879	0.762	0.091
LSA, $K = 15$	0.958	0.951	0.939	0.927	0.906	0.792	0.085
LSA, $K = 30$	0.966	0.958	0.946	0.932	0.908	0.779	0.095
LSA, $K = 50$	0.961	0.952	0.936	0.920	0.890	0.756	0.107
PLSA-1, $K = 5$	0.862	0.862	0.858	0.850	0.840	0.800	0.047
PLSA-1, $K = 7$	0.938	0.936	0.931	0.925	0.916	0.863	0.037
PLSA-1, $K = 15$	0.964	0.958	0.950	0.942	0.928	0.824	0.063
PLSA-1, $K = 30$	0.968	0.962	0.952	0.942	0.924	0.773	0.093
PLSA-1, $K = 50$	0.967	0.959	0.946	0.935	0.914	0.751	0.105
PLSA-2, $K = 5$	0.866	0.864	0.858	0.849	0.835	0.767	0.059
PLSA-2, $K = 7$	0.940	0.936	0.930	0.920	0.907	0.841	0.040
PLSA-2, $K = 15$	**0.973**	**0.969**	**0.961**	**0.952**	0.934	0.809	0.070
PLSA-2, $K = 30$	**0.974**	**0.970**	**0.960**	0.949	0.927	0.760	0.100
PLSA-2, $K = 50$	**0.975**	**0.969**	0.957	0.944	0.916	0.740	0.108
AFC, $K = 5$	0.941	0.939	0.936	0.931	0.916	0.863	**0.034**
AFC, $K = 7$	0.959	0.955	0.948	0.942	0.933	**0.873**	**0.034**
AFC, $K = 15$	0.970	**0.965**	0.955	0.948	**0.936**	0.837	0.049
AFC, $K = 30$	**0.971**	**0.965**	0.954	0.945	0.930	0.812	0.059
AFC, $K = 50$	0.968	0.962	0.951	0.941	0.922	0.795	0.065

TABLE 3.5 – Résultats des différentes méthodes sur la base Caltech-101.

Méthodes ↓	P@10	P@20	P@50	P@100	P@200	MAP	$MANR$
*tf*idf*	0.258	0.244	0.217	0.192	0.166	0.136	0.295
*tf*idf*-1	0.259	0.246	0.219	0.194	0.168	0.137	0.294
*tf*idf*-2	0.248	0.234	0.208	0.184	0.158	0.129	0.299
LSA, $K = 30$	0.310	0.288	0.257	0.231	0.202	0.151	0.278
LSA, $K = 50$	0.323	0.298	0.263	0.233	0.202	0.154	0.277
LSA, $K = 75$	0.326	0.300	0.263	0.231	0.199	0.153	0.278
LSA, $K = 100$	0.327	0.300	0.261	0.228	0.195	0.152	0.280
LSA, $K = 200$	0.317	0.290	0.249	0.214	0.181	0.146	0.287
PLSA-1, $K = 30$	0.294	0.277	0.252	0.231	0.210	0.174	0.262
PLSA-1, $K = 50$	0.326	0.307	0.279	0.252	0.227	0.195	0.253
PLSA-1, $K = 75$	0.333	0.312	0.283	0.255	0.225	0.186	0.257
PLSA-1, $K = 100$	0.333	0.312	0.280	0.251	0.221	0.181	0.259
PLSA-1, $K = 200$	0.337	0.317	0.286	0.255	0.223	0.185	0.261
PLSA-2, $K = 30$	0.307	0.285	0.254	0.227	0.197	0.158	0.267
PLSA-2, $K = 50$	0.336	0.312	0.277	0.245	0.212	0.173	0.260
PLSA-2, $K = 75$	0.337	0.312	0.277	0.244	0.209	0.169	0.264
PLSA-2, $K = 100$	0.330	0.305	0.268	0.234	0.199	0.160	0.270
PLSA-2, $K = 200$	0.313	0.289	0.256	0.224	0.191	0.155	0.276
AFC, $K = 30$	0.330	0.309	0.277	0.249	0.220	0.182	0.250
AFC, $K = 50$	0.360	0.336	0.300	0.268	**0.236**	**0.198**	**0.240**
AFC, $K = 75$	**0.369**	0.343	**0.303**	**0.269**	0.234	0.197	**0.240**
AFC, $K = 100$	**0.371**	**0.344**	0.302	0.266	0.230	0.194	**0.241**
AFC, $K = 200$	0.367	0.339	0.295	0.257	0.220	0.185	0.246

TABLE 3.6 – Résultats de différentes méthodes sur la base Nistér-Stewénius.

Méthodes ↓	Score N-S	P@3	MAP	$MANR$
*tf*idf*	2.767	0.589	0.626	0.018
*tf*idf*-1	2.743	0.581	0.619	0.019
*tf*idf*-2	2.747	0.582	0.620	0.022
LSA, $K = 100$	2.891	0.630	0.670	0.015
LSA, $K = 200$	2.973	0.658	0.696	0.015
LSA, $K = 300$	2.995	0.665	0.701	0.015
LSA, $K = 400$	2.989	0.663	0.698	0.016
LSA, $K = 500$	2.980	0.660	0.695	0.017
PLSA-1, $K = 100$	2.825	0.608	0.651	0.010
PLSA-1, $K = 200$	2.833	0.611	0.656	0.011
PLSA-1, $K = 300$	2.745	0.582	0.628	0.011
PLSA-1, $K = 400$	2.706	0.569	0.613	0.012
PLSA-1, $K = 500$	2.631	0.544	0.591	0.014
PLSA-2, $K = 100$	3.122	0.707	0.742	0.009
PLSA-2, $K = 200$	3.154	0.718	0.762	0.009
PLSA-2, $K = 300$	3.116	0.705	0.753	0.010
PLSA-2, $K = 400$	3.059	0.686	0.740	0.011
PLSA-2, $K = 500$	3.061	0.687	0.738	0.012
AFC, $K = 100$	3.123	0.708	0.747	0.007
AFC, $K = 200$	3.195	0.732	0.770	**0.006**
AFC, $K = 300$	**3.217**	**0.739**	0.776	**0.006**
AFC, $K = 400$	**3.222**	**0.741**	0.777	**0.006**
AFC, $K = 500$	**3.225**	**0.742**	**0.778**	**0.006**
AFC, $K = 600$	**3.225**	**0.742**	**0.778**	**0.006**

TABLE 3.7 – Résultats de l'AFC avec différentes mesures de similarité sur les bases Catech-4 et Caltech-101.

Bases/Méthodes ↓		P@10	P@20	P@50	P@100	P@200	*MAP*	*MANR*
Caltech-4 $K = 7$	L_1	0.961	0.954	0.944	0.933	0.917	0.794	0.081
	L_2	**0.962**	**0.956**	0.946	0.935	0.920	0.800	0.081
	Cosinus	0.959	**0.955**	**0.948**	**0.942**	**0.933**	**0.873**	**0.034**
Caltech-101 $K = 50$	L_1	0.335	0.307	0.267	0.234	0.197	0.140	0.321
	L_2	0.341	0.313	0.272	0.239	0.203	0.149	0.313
	Cosinus	**0.360**	**0.336**	**0.300**	**0.268**	**0.236**	**0.198**	**0.240**

TABLE 3.8 – Résultats de l'AFC avec différentes mesures de similarité sur la base Nistér-Stewénius.

Bases/Méthodes ↓		Score N-S	P@3	*MAP*	*MANR*
Nistér-Stewénius $K = 200$	L_1	2.943	0.648	0.677	0.056
	L_2	2.992	0.664	0.694	0.043
	Cosinus	**3.195**	**0.732**	**0.770**	**0.006**

bien catégorisées), le PLSA fait de moins en moins bien que l'AFC (cf. les résultats sur les bases Caltech-101 et Nistér-Stewénius). La mesure J-S donne de meilleurs résultats que la similarité du cosinus dans le cas du PLSA. Cependant le temps de calcul de la mesure J-S est beaucoup plus élevé que celui de la similarité du cosinus à cause du calcul des logarithmes (environ 60 fois plus long).

Dans les bases Caltech-4 et Caltech-101, les images sont catégorisées. La variance intra-classe est assez élevée. Une petite valeur de K (par ex. 7 et 50 respectivement) donne de meilleurs résultats. Tandis que les images dans un groupe de la base Nistér-Stewénius sont très ressemblantes. Les meilleurs résultats sont obtenus avec un grand K (par ex. 500).

AFC avec différentes mesures de similarité

Dans les expérimentations qui suivent, nous comparons les résultats de l'AFC obtenus avec la distance L_1, L_2 et la similarité du cosinus. Les résultats apparaissent dans la figure 3.9 et les tableaux 3.7 et 3.8.

Dans tous les cas, la similarité du cosinus donne de meilleurs résultats et la distance L_2 est la deuxième.

Temps de réponse

Le tableau 3.9 montre le temps de réponse moyen (en millisecondes), pour une requête, des méthodes PLSA, AFC et *tf*idf* (sans et avec fichiers inversés). La technique de fichiers inversés a été utilisée pour accélérer la recherche dans le cas de *tf*idf*. Pour le PLSA et l'AFC, les représentations des images ne sont plus creuses, la recherche

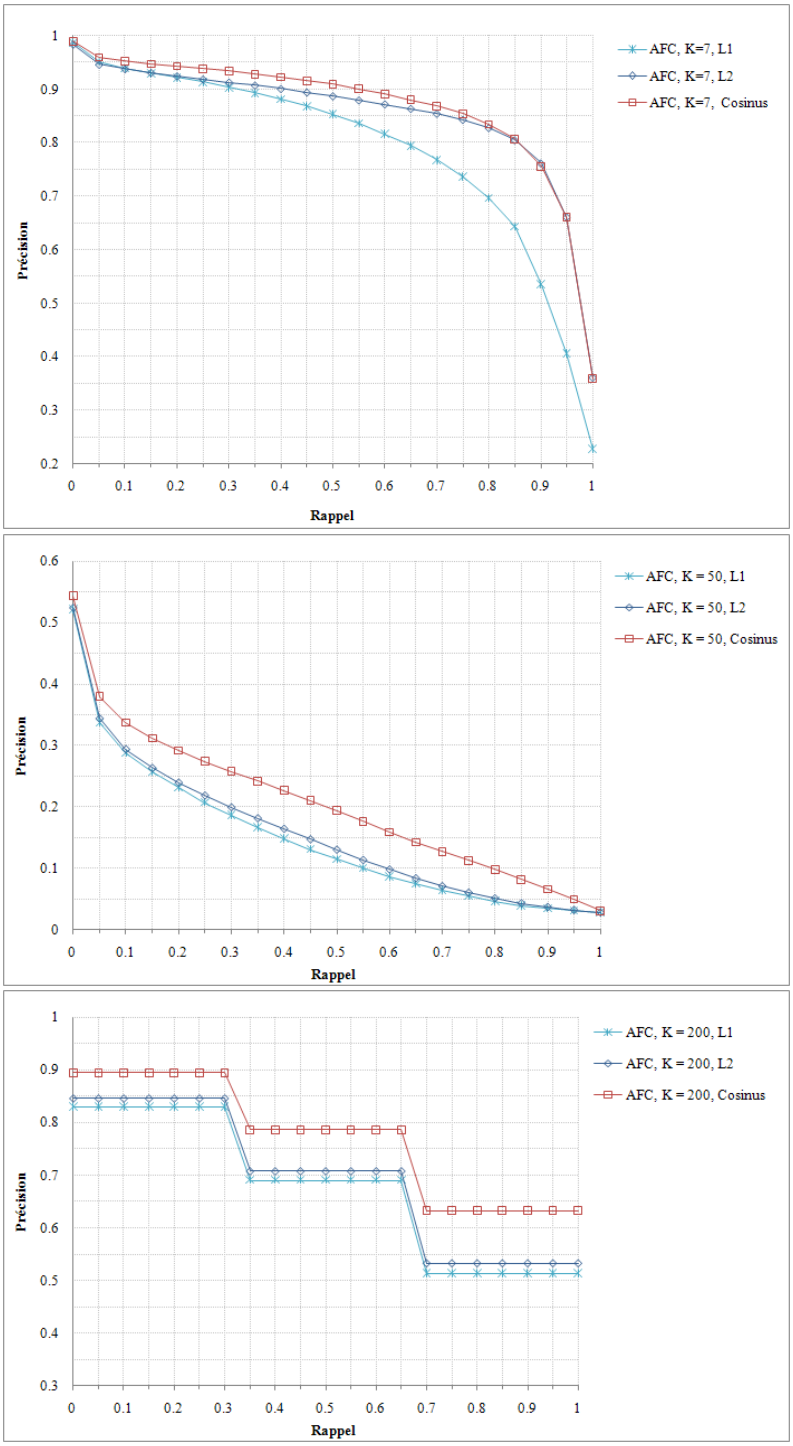


FIGURE 3.9 – Courbes de *précision–rappel* de l'AFC avec différentes mesures de similarité.

TABLE 3.9 – Temps de réponse moyen pour une requête (en millisecondes). *tf*idf* (1) : *tf*idf* sans fichiers inversés, *tf*idf* (2) : *tf*idf* avec fichiers inversés.

Bases ↓	AFC/PLSA	*tf*idf* (1)	*tf*idf* (2)
Caltech-4 ($K = 7$)	**0.385**		
Caltech-4 ($K = 15$)	0.434	25.042	0.962
Caltech-4 ($K = 30$)	0.522		
Caltech-101 ($K = 50$)	3.319		
Caltech-101 ($K = 100$)	4.682	121.798	**2.831**
Caltech-101 ($K = 200$)	7.125		
Nistér-Stewénius ($K = 100$)	**5.554**		
Nistér-Stewénius ($K = 200$)	8.354	140.459	13.716
Nistér-Stewénius ($K = 500$)	16.504		

séquentielle est donc utilisée. En général, la réduction de dimensions effectuée par les méthodes PLSA et AFC réduit environ 2 fois le temps de réponse par rapport à la méthode *tf*idf* avec fichiers inversés. Cependant, le gain d'accélération des méthodes PLSA et AFC par rapport à la méthode *tf*idf* avec fichiers inversés dépend du nombre d'éléments non nuls des tableaux de contingence. Plus ce nombre est grand, plus le gain est grand. En particulier, dans le cas de la base Caltech-101, la technique des fichiers inversés est plus rapide qu'une recherche séquentielle sur les représentations réduites par le PLSA ou l'AFC car le tableau de contingence est très creux.

3.6.9 Recherche approximative à l'aide des fichiers inversés

Puisque la représentation réduite des images issue de l'AFC n'est plus creuse, une recherche séquentielle doit être utilisée pour faire une recherche exacte. Nous présentons dans cette section, les expérimentations de la recherche approximative des images à l'aide des fichiers inversés basées sur les indicateurs de l'AFC. Pour évaluer la performance des méthodes, nous calculons la précision à certaines premières images retournées. Le gain d'accélération des méthodes approximatives par rapport à la méthode exhaustive figure aussi dans les résultats.

Les tableaux 3.10 et 3.11 présentent les résultats expérimentaux de la recherche approximative à l'aide des fichiers inversés sur les indicateurs issus de l'AFC. "*Approx. 1*" désigne la méthodes avec des fichiers inversés basés sur la contribution et la méthode "*Approx. 2*" se base sur la qualité de représentation. La méthode exhaustive parcourt toute la base d'images tandis que les méthodes approximatives s'exécutent en 2 étapes : (i) filtrer les images non pertinentes à l'aide des fichiers inversés et (ii) raffiner les résultats par une recherche séquentielle dans la liste d'images candidates $\mathcal{H}_{\hat{\theta}}$. Le seuil $\hat{\theta}$ est choisi tel que la liste d'images candidates contient au moins 500 images. Le temps de réponse moyens des méthodes utilisées est présentés dans le tableau 3.12. Pour les méthodes approximatives, nous montrons le temps de calcul pour l'étape de filtrage et l'étape de raffinement dans les deux colonnes "*Filtrage*" et "*Raffinement*" respec-

TABLE 3.10 – Résultats de la recherche approximative avec fichiers inversés sur les bases Catech-4 et Caltech-101.

Bases/Méthodes ↓		P@10	P@20	P@50	P@100	MAP
Caltech-4 $K = 7$	Exhaustive	0.959	0.955	0.948	**0.942**	**0.515**
	Approx. 1	**0.964**	**0.959**	**0.951**	0.940	0.475
	Approx. 2	0.959	0.955	0.948	**0.941**	0.506
Caltech-4 $K = 15$	Exhaustive	**0.970**	**0.965**	**0.955**	**0.948**	**0.499**
	Approx. 1	**0.969**	**0.963**	**0.953**	0.941	0.435
	Approx. 2	**0.970**	**0.964**	**0.954**	**0.948**	0.492
Caltech-4 $K = 30$	Exhaustive	**0.971**	**0.965**	**0.954**	**0.945**	**0.484**
	Approx. 1	0.969	0.962	0.948	0.933	0.369
	Approx. 2	**0.971**	**0.964**	0.953	0.944	0.466
Caltech-101 $K = 50$	Exhaustive	**0.360**	**0.336**	**0.300**	**0.268**	**0.144**
	Approx. 1	0.356	0.331	0.293	0.257	0.105
	Approx. 2	**0.360**	**0.336**	0.299	0.267	0.131
Caltech-101 $K = 100$	Exhaustive	0.371	0.344	0.302	**0.266**	**0.141**
	Approx. 1	0.361	0.331	0.282	0.230	0.081
	Approx. 2	**0.371**	**0.345**	**0.302**	**0.266**	0.130
Caltech-101 $K = 200$	Exhaustive	0.367	0.339	0.295	0.257	0.134
	Approx. 1	0.346	0.312	0.248	0.179	0.061
	Approx. 2	**0.369**	**0.341**	**0.296**	**0.258**	0.118

tivement. Le gain d'accélération des méthodes approximatives est aussi figuré dans la colonne "*Gain*". La dernière colonne, $|\mathcal{H}_{\hat{\theta}}|$, présente la taille moyenne de la liste d'images candidates.

La méthode basée sur la contribution accélère la recherche de 2.66 à 17.25 et la méthode basée sur la qualité de représentation est de 3.18 à 13.78 fois plus rapide que la recherche séquentielle. En ce qui concerne la pertinence des images retournées (la *précision*), la méthode basée sur la qualité de représentation fait aussi bien, voire mieux dans certains cas (cf. Table 3.11), que la méthode exhaustive tandis que la méthode basée sur la contribution fournit de moins bons résultats. Dans le cas le pire, elle dégrade la *précision* d'environ 10% (cf. Table 3.11) par rapport à la méthode exhaustive mais est encore meilleure que la pondération *tf*idf*. Ceci peut être expliqué par le fait que les nouvelles images ne participent pas à la construction des axes factoriels, elles ont une contribution nulle (nous avons utilisé la *pseudo* contribution à la place de la contribution). Par ailleurs, le seuil pour déterminer les thèmes associés à une requête dépend des images qui participent à la construction des axes ($\varepsilon = \frac{1}{M}$ avec M est le nombre de ces images). Ce n'est pas le cas pour la méthode basée sur la qualité de représentation où le seuil ne dépend de que la requête ($\varepsilon = \frac{1}{K}$ avec K est le nombre d'axes conservés).

Pour étudier l'impact du paramètre ε à la performance de la méthode de recherche basée sur les fichiers inversés, nous avons varié ce paramètre avec les valeurs $\frac{2}{K}$, $\frac{1}{K}$, $\frac{1}{2K}$, $\frac{1}{4K}$, $\frac{1}{8K}$ et 0. Les résultats sont présentés dans les tableaux 3.13 et 3.14. Dans ces

TABLE 3.11 – Résultats de la recherche approximative avec fichiers inversés sur la base Nistér-Stewénius.

Bases/Méthodes ↓		P@3	P@10	P@20	P@50	P@100	MAP
$K = 100$	Exhaustive	**0.708**	**0.240**	**0.126**	**0.053**	**0.027**	**0.747**
	Approx. 1	0.611	0.199	0.103	0.042	0.021	0.629
	Approx. 2	0.707	0.240	0.126	**0.053**	0.027	0.745
$K = 200$	Exhaustive	**0.732**	0.246	**0.128**	**0.054**	**0.028**	**0.770**
	Approx. 1	0.625	0.201	0.103	0.042	0.021	0.640
	Approx. 2	**0.732**	**0.246**	**0.128**	**0.054**	**0.028**	0.769
$K = 500$	Exhaustive	0.742	0.248	0.129	0.054	0.028	0.778
	Approx. 1	0.647	0.206	0.104	0.042	0.021	0.659
	Approx. 2	**0.748**	**0.251**	**0.131**	**0.055**	**0.029**	**0.785**

deux tableaux, la colonne "*Filtrage*" désigne le temps de calcul moyen (en milliseconde) pour l'étape de filtrage et la colonne "*Total*" montre le temps de réponse moyen pour une requête. La taille moyenne des listes d'images candidates est présentée dans la dernière colonne. Les meilleurs MAP sont obtenus avec ε dans l'intervalle $[\frac{1}{K}; \frac{1}{4K}]$. Plus le paramètre ε est petit, plus les fichiers inversés sont gros. Par conséquent, le temps de filtrage est grand. De plus, si ε est trop petit (par ex., $\varepsilon = 0$), les fichiers inversés peuvent être contenir des images n'ayant une grand qualité de représentation. Les résultats deviennent moins bons. Le choix de ce paramètre égal à $\frac{1}{K}$ est un bon compromis entre la pertinence des résultats et la rapidité.

3.7 Synthèse

Après avoir décrit brièvement l'Analyse Factorielle des Correspondances, nous avons présenté son application aux images. Dans ce domaine, l'AFC permet une représentation des images à un niveau sémantique plus élevé que la représentation originale basée sur la fréquence des mots visuels et réduit ainsi le fossé sémantique dans la description du contenu des images. Les expérimentations réalisées sur les 3 bases d'images ont montré que l'AFC fait mieux que le PLSA et beaucoup mieux que la pondération *tf*idf* en termes de pertinence des images retournées.

La réduction de dimensions par l'AFC ou le PLSA améliore le temps de réponse (par rapport à la technique des fichiers inversés appliquée directement aux tableaux de contingence) dans le cas où le nombre d'axes conservés est petit et les tableaux de contingence ne sont pas très creux.

Nous avons aussi proposé une algorithme de recherche approximative en deux étapes pour accélérer la recherche sans trop diminuer la qualité de résultats. La première étape consiste à filtrer les images non pertinentes à l'aide des fichiers inversés basés sur les indicateurs de l'AFC : la *contribution* et la *qualité de représentation*. Les résultats expérimentaux montrent que les méthodes approximatives sont de 2.66 à 17.25 fois plus rapides que la méthode exhaustive. Quant à la pertinence des images retournées,

TABLE 3.12 – Temps de réponse moyen pour la méthode exhaustive et les méthodes approximatives.

Bases/Méthodes ↓		Temps de réponse			Gain	$\lvert\mathcal{H}_{\hat{\theta}}\rvert$
		Filtrage	Raffinement	Total		
Caltech-4 $K = 7$	Exhaustive	-	-	0.385	-	-
	Approx. 1	0.003	0.142	0.145	2.66	1 506
	Approx. 2	0.006	0.116	**0.121**	**3.18**	1 239
Caltech-4 $K = 15$	Exhaustive	-	-	0.434	-	-
	Approx. 1	0.005	0.134	0.139	3.13	1 242
	Approx. 2	0.010	0.113	**0.123**	**3.53**	1 056
Caltech-4 $K = 30$	Exhaustive	-	-	0.522	-	-
	Approx. 1	0.009	0.134	0.144	3.63	1 025
	Approx. 2	0.017	0.116	**0.133**	**3.93**	890
Caltech-101 $K = 50$	Exhaustive	-	-	3.319	-	-
	Approx. 1	0.022	0.393	0.415	8.00	1 488
	Approx. 2	0.065	0.176	**0.241**	**13.78**	810
Caltech-101 $K = 100$	Exhaustive	-	-	4.682	-	-
	Approx. 1	0.059	0.285	**0.344**	**13.63**	853
	Approx. 2	0.120	0.260	0.380	12.33	714
Caltech-101 $K = 200$	Exhaustive	-	-	7.125	-	-
	Approx. 1	0.167	0.334	**0.501**	**14.22**	636
	Approx. 2	0.227	0.433	0.659	10.81	655
Nistér-Stewénius $K = 100$	Exhaustive	-	-	5.554	-	-
	Approx. 1	0.077	0.245	**0.322**	**17.25**	774
	Approx. 2	0.132	0.271	0.402	13.81	769
Nistér-Stewénius $K = 200$	Exhaustive	-	-	8.354	-	-
	Approx. 1	0.206	0.309	**0.515**	**16.23**	615
	Approx. 2	0.238	0.419	0.657	12.71	684
Nistér-Stewénius $K = 500$	Exhaustive	-	-	16.504	-	-
	Approx. 1	0.691	0.712	**1.403**	**11.76**	557
	Approx. 2	0.583	0.905	1.488	11.09	617

TABLE 3.13 – Impact du paramètre ε à la méthode de recherche approximative basée sur la qualité de représentation : Résultats sur les bases Caltech-4 et Caltech-101.

Bases/Méthodes ↓		P@10	P@20	P@50	P@100	MAP	Filtrage	Total	$\lvert\mathcal{H}_{\hat{\theta}}\rvert$
	$\varepsilon = 2/K$	0.957	0.951	0.939	0.911	0.410	0.002	0.074	782
	$\varepsilon = 1/K$	0.959	**0.955**	**0.948**	0.941	**0.506**	0.006	0.121	1 239
Caltech-4	$\varepsilon = 1/(2K)$	**0.960**	**0.955**	**0.948**	**0.942**	**0.510**	0.011	0.107	1 038
$K = 7$	$\varepsilon = 1/(4K)$	**0.959**	**0.955**	**0.949**	0.942	**0.505**	0.018	0.107	951
	$\varepsilon = 1/(8K)$	**0.959**	**0.955**	0.948	0.941	0.495	0.024	0.108	909
	$\varepsilon = 0$	0.959	**0.955**	0.947	0.941	0.482	0.044	0.124	879
	$\varepsilon = 2/K$	**0.970**	**0.965**	**0.955**	0.943	0.482	0.004	0.121	1 110
	$\varepsilon = 1/K$	**0.970**	0.964	**0.955**	**0.948**	**0.492**	0.010	0.123	1 056
Caltech-4	$\varepsilon = 1/(2K)$	**0.970**	**0.965**	0.955	0.947	0.476	0.018	0.112	889
$K = 15$	$\varepsilon = 1/(4K)$	**0.970**	0.964	0.954	0.945	0.450	0.030	0.116	798
	$\varepsilon = 1/(8K)$	0.969	0.963	0.954	0.944	0.432	0.040	0.121	759
	$\varepsilon = 0$	0.969	0.964	0.953	0.943	0.394	0.087	0.164	708
	$\varepsilon = 2/K$	**0.971**	**0.965**	**0.955**	**0.946**	**0.485**	0.007	0.157	1 163
	$\varepsilon = 1/K$	**0.971**	0.964	0.953	0.943	0.466	0.017	0.133	890
Caltech-4	$\varepsilon = 1/(2K)$	0.971	0.964	0.953	0.942	0.419	0.034	0.135	775
$K = 30$	$\varepsilon = 1/(4K)$	0.971	0.964	0.953	0.941	0.380	0.056	0.150	716
	$\varepsilon = 1/(8K)$	**0.971**	0.964	0.953	0.940	0.353	0.084	0.173	683
	$\varepsilon = 0$	0.970	0.964	0.953	0.938	0.320	0.168	0.257	653
	$\varepsilon = 2/K$	0.358	0.334	0.297	0.264	0.127	0.023	0.262	1 100
	$\varepsilon = 1/K$	**0.360**	**0.336**	0.299	0.267	**0.131**	0.065	0.241	810
Caltech-101	$\varepsilon = 1/(2K)$	**0.360**	**0.336**	**0.300**	0.267	**0.132**	0.136	0.288	704
$K = 50$	$\varepsilon = 1/(4K)$	**0.360**	**0.336**	**0.299**	**0.267**	**0.131**	0.232	0.378	664
	$\varepsilon = 1/(8K)$	**0.359**	0.335	**0.299**	**0.267**	**0.129**	0.332	0.476	644
	$\varepsilon = 0$	**0.360**	0.335	0.298	0.265	0.124	0.722	0.877	621
	$\varepsilon = 2/K$	0.370	0.343	0.300	0.264	0.129	0.041	0.359	894
	$\varepsilon = 1/K$	**0.371**	**0.345**	**0.302**	**0.266**	**0.130**	0.120	0.380	714
Caltech-101	$\varepsilon = 1/(2K)$	**0.372**	**0.344**	**0.301**	**0.265**	0.124	0.262	0.500	647
$K = 100$	$\varepsilon = 1/(4K)$	**0.371**	**0.344**	**0.300**	0.264	0.119	0.447	0.683	618
	$\varepsilon = 1/(8K)$	**0.371**	0.343	0.299	0.262	0.114	0.647	0.885	601
	$\varepsilon = 0$	0.370	0.343	0.297	0.259	0.108	1.429	1.710	588
	$\varepsilon = 2/K$	0.368	0.340	0.295	0.257	**0.124**	0.077	0.566	777
	$\varepsilon = 1/K$	**0.369**	**0.341**	**0.296**	**0.258**	0.118	0.227	0.659	655
Caltech-101	$\varepsilon = 1/(2K)$	**0.369**	**0.340**	**0.294**	0.254	0.107	0.417	0.938	610
$K = 200$	$\varepsilon = 1/(4K)$	**0.369**	**0.338**	0.290	0.249	0.097	0.903	1.333	587
	$\varepsilon = 1/(8K)$	**0.368**	0.337	0.288	0.246	0.092	1.310	1.750	576
	$\varepsilon = 0$	0.366	0.337	0.286	0.244	0.089	2.961	3.450	565

TABLE 3.14 – Impact du paramètre ε à la méthode de recherche approximative basée sur la qualité de représentation : Résultats sur la base Nistér.

Bases/Méthodes ↓		P@3	P@10	P@20	P@50	P@100	MAP	Filtrage	Total	$\lvert\mathcal{H}_{\hat{\theta}}\rvert$
	$\varepsilon = 2/K$	0.703	0.238	0.124	0.052	0.027	0.739	0.058	0.381	950
	$\varepsilon = 1/K$	**0.707**	**0.240**	**0.126**	**0.053**	**0.027**	0.745	0.132	0.402	769
Nister	$\varepsilon = 1/(2K)$	0.707	**0.240**	**0.126**	**0.053**	**0.027**	0.746	0.271	0.510	671
$K = 100$	$\varepsilon = 1/(4K)$	**0.708**	**0.241**	**0.126**	**0.053**	**0.027**	**0.747**	0.478	0.713	630
	$\varepsilon = 1/(8K)$	0.708	**0.241**	**0.126**	**0.053**	**0.027**	**0.747**	0.708	0.943	613
	$\varepsilon = 0$	0.707	0.240	0.126	0.53	0.027	0.746	1.722	2.000	591
	$\varepsilon = 2/K$	0.729	0.245	0.127	0.053	0.027	0.765	0.101	0.576	807
	$\varepsilon = 1/K$	0.732	0.246	0.128	**0.054**	**0.028**	0.769	0.238	0.657	684
Nister	$\varepsilon = 1/(2K)$	**0.734**	0.247	**0.129**	**0.054**	**0.028**	0.772	0.530	0.927	621
$K = 200$	$\varepsilon = 1/(4K)$	**0.735**	**0.248**	**0.129**	**0.054**	**0.028**	**0.773**	0.954	1.353	593
	$\varepsilon = 1/(8K)$	**0.735**	**0.248**	**0.129**	**0.054**	**0.028**	**0.773**	1.406	1.816	581
	$\varepsilon = 0$	0.731	0.246	0.128	0.054	0.028	0.769	3.558	4.035	566
	$\varepsilon = 2/K$	0.742	0.248	0.129	0.054	0.028	0.777	0.213	1.171	693
	$\varepsilon = 1/K$	0.748	0.251	0.131	**0.055**	**0.028**	0.785	0.583	1.488	617
Nister	$\varepsilon = 1/(2K)$	0.758	**0.254**	**0.132**	**0.055**	**0.028**	0.794	1.348	2.252	578
$K = 500$	$\varepsilon = 1/(4K)$	**0.760**	**0.255**	**0.132**	**0.055**	0.028	**0.796**	2.496	3.425	560
	$\varepsilon = 1/(8K)$	**0.760**	**0.254**	**0.132**	0.054	0.028	0.794	3.766	4.714	553
	$\varepsilon = 0$	0.740	0.248	0.129	0.054	0.028	0.776	9.410	10.283	544

la méthode basée sur la *qualité de représentation* fait aussi bien, voire mieux qu'une recherche séquentielle tandis que la méthode utilisant la *contribution* fournit de moins bons résultats.

Chapitre 4

Passage à l'échelle

Nous avons montré dans le chapitre précédent que l'application de l'AFC aux images améliore considérablement la qualité de la recherche d'images par le contenu utilisant un modèle « sac-de-mots-visuels ». Dans ce chapitre, nous étudions le passage à l'échelle de l'AFC afin de pouvoir traiter de grandes bases d'images.

Nous présentons d'abord le calcul de l'AFC pour de grands tableaux de contingence, puis les méthodes pour l'accélération de la recherche d'images à l'aide des fichiers inversés basés sur l'AFC et la combinaison de l'AFC avec d'autres méthodes pour améliorer la qualité de recherche.

4.1 AFC pour les grands tableaux de contingence

4.1.1 AFC incrémentale

Rappelons que le calcul de l'AFC consiste à construire et à chercher les valeurs propres et les vecteurs propres d'une matrice particulière [1]

$$\mathbf{A} = \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1}$$

où $\mathbf{F}$ est la matrice de données (le tableau de fréquences relatives) ; $\mathbf{P}$ et $\mathbf{Q}$ sont des matrices diagonales dont les éléments diagonaux sont les totaux marginaux en ligne et en colonne respectivement de la matrice $\mathbf{F}$ (cf. Table 3.1 dans le cas de l'analyse dans l'espace $\mathbb{R}^N$).

Pour les grands tableaux de contingence, il est impossible de charger la matrice de données $\mathbf{F}$ en mémoire. Nous proposons donc une procédure incrémentale pour construire la matrice $\mathbf{A}$. Cette méthode incrémentale conduit à une matrice $\mathbf{A}$ identique à celle que l'on obtient dans la méthode non incrémentale. En particulier, nous obtenons les mêmes valeurs propres et les mêmes vecteurs propres.

1. Pour l'implémentation, nous avons plutôt utilisé la matrice $\mathbf{T} = \hat{\mathbf{Q}}^{-\frac{1}{2}}\mathbf{X}^{\mathrm{T}}\hat{\mathbf{P}}^{-1}\mathbf{X}\hat{\mathbf{Q}}^{-\frac{1}{2}}$ calculée à partir du tableau de contingence brut $\mathbf{X}$ au lieu du tableau des fréquences relatives $\mathbf{F}$ (cf. Formule 3.46). La projection des images est aussi effectuée de la même manière (cf. Formule 3.53). Il suffit de remplacer $\mathbf{F}$ par $\mathbf{X}$ et faire quelques petites modifications.

Si on découpe la matrice $\mathbf{F}$ en B blocs par ligne :

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_{[1]} \\ \mathbf{F}_{[2]} \\ \vdots \\ \mathbf{F}_{[B]} \end{pmatrix},$$

les totaux marginaux correspondant à chaque profil-ligne (les éléments diagonaux de la matrices $\mathbf{P}$) peuvent être calculés indépendamment pour chaque bloc $\mathbf{F}_{[i]}$ de la matrice $\mathbf{F}$. On obtient alors, pour chaque bloc de données $\mathbf{F}_{[i]}$, une matrice $\mathbf{P}_{[i]}$ qui contient les totaux marginaux des profils-lignes du bloc en examen. La matrice $\mathbf{P}$ ainsi s'exprime par :

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{[1]} & & & 0 \\ & \mathbf{P}_{[2]} & & \\ & & \ddots & \\ 0 & & & \mathbf{P}_{[B]} \end{pmatrix} \tag{4.1}$$

L'inversion de $\mathbf{P}$ est obtenue par :

$$\mathbf{P}^{-1} = \begin{pmatrix} \left(\mathbf{P}_{[1]}\right)^{-1} & & & 0 \\ & \left(\mathbf{P}_{[2]}\right)^{-1} & & \\ & & \ddots & \\ 0 & & & \left(\mathbf{P}_{[B]}\right)^{-1} \end{pmatrix} \tag{4.2}$$

car $\mathbf{P}$ est une matrice diagonale.

Par contre, les totaux marginaux de chaque profil-colonne doivent être calculés sur tous les blocs de données. Si on note $\mathbf{Q}_{[i]}$ la matrice contenant les totaux marginaux de chaque profil-colonnes calculés sur le bloc $\mathbf{F}_{[i]}$, la matrice $\mathbf{Q}$ s'obtient par la somme de toutes les $\mathbf{Q}_{[i]}$, c'est à dire :

$$\mathbf{Q} = \sum_{i=1}^{B} \mathbf{Q}_{[i]} \tag{4.3}$$

Si on note $\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}$, $\mathbf{A}$ s'exprime alors par :

$$\begin{aligned} \mathbf{A} &= \mathbf{F}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{F}\mathbf{Q}^{-1} \\ &= \mathbf{C}\mathbf{Q}^{-1} \end{aligned} \tag{4.4}$$

La matrice $\mathbf{C}$ est calculée de façon incrémentale sur tous les blocs $\mathbf{F}_{[i]}$:

$$\mathbf{C} = \sum_{i=1}^{B} \mathbf{C}_{[i]} \tag{4.5}$$

avec

$$\forall i \in [1;B], \mathbf{C}_{[i]} = \mathbf{F}_{[i]}^{\mathrm{T}} \mathbf{P}_{[i]}^{-1} \mathbf{F}_{[i]} \tag{4.6}$$

Les deux formules 4.3 et 4.5 nous aident à construire un algorithme incrémental pour calculer $\mathbf{A}$. La projection des images $\mathbf{Z}$ sur les axes factoriels (cf. Formule 3.23) peut être réalisée d'une manière analogue :

$$\mathbf{Z} = \begin{pmatrix} \mathbf{Z}_{[1]} \\ \mathbf{Z}_{[2]} \\ \vdots \\ \mathbf{Z}_{[B]} \end{pmatrix} \tag{4.7}$$

où

$$\forall i \in [1;B], \mathbf{Z}_{[i]} = \mathbf{P}_{[i]}^{-1} \mathbf{F}_{[i]} \mathbf{Q}^{-1} \mathbf{U} \tag{4.8}$$

La procédure incrémentale pour calculer l'AFC est décrite dans l'algorithme 3. D'abord, on calcule $\mathbf{Q}$ et $\mathbf{C}$ de façon incrémentale (lignes 1–7). Puis la matrice $\mathbf{A}$ s'obtient à partir de $\mathbf{Q}$ et $\mathbf{C}$ (ligne 8). Après avoir obtenu les valeurs propres de $\mathbf{A}$, on calcule (de façon incrémentale aussi) la projection des images (lignes 10–13).

Algorithme 3 : Algorithme incrémental pour calculer l'AFC

```
1  Q = 0
2  C = 0
3  pour i = 1 à B faire
4  |  charger le bloc F_[i] en mémoire
5  |  calculer P_[i], Q_[i] et C_[i] à partir de F_[i]
6  |  Q = Q + Q_[i]
7  |  C = C + A_[i]
8  A = CQ^{-1}
9  calculer K premiers vecteurs propres de A
10 pour i = 1 à B faire
11 |  charger le bloc F_[i] en mémoire
12 |  calculer P_[i] à partir de F_[i]
13 |  calculer la projection Z_[i] = P_[i]^{-1} F_[i] Q^{-1} U
```

4.1.2 AFC incrémentale parallèle

L'algorithme incrémental présenté ci-dessus permet de traiter de grandes bases d'images. Cependant, il est exécuté de manière séquentielle sur une seule machine. Comme les matrices $\mathbf{Q}_{[i]}$ et $\mathbf{C}_{[i]}$ sont calculée indépendamment pour chaque bloc de données $\mathbf{F}_{[i]}$, il est possible de répartir les blocs de données sur plusieurs machines. Les matrices $\mathbf{Q}_{[i]}$ et $\mathbf{C}_{[i]}$ sont calculées indépendamment sur chaque machine. Lorsque toutes les machines finissent leurs tâches, les résultats individuels sont combinés pour obtenir un résultat final.

Au cours des années passées, le GPU (Graphics Processing Unit) a évolué et a offert des ressources incroyables pour les traitements graphiques et non-graphiques. Le GPU est spécialisé pour des calculs intensifs et hautement parallèles. En 2006, NVIDIA a introduit CUDA (Compute Unified Device Architecture) [NBGS08, NVI08b], une architecture de calcul parallèle "general purpose" - avec un nouveau modèle de programmation et un ensemble d'instructions - qui tire profit du moteur de calcul parallèle du GPU NVIDIA pour résoudre de nombreux problèmes de calcul complexes de façon beaucoup plus rapide que sur un CPU. CUDA fournit un environnement de programmation qui permet aux programmeurs d'utiliser le langage de programmation C comme un langage de programmation de haut niveau. Dans l'architecture CUDA, le GPU est un périphérique qui peut exécuter plusieurs "*threads*" concurrents.

Une implémentation de BLAS (Basic Linear Algebra Subprograms) sur NVIDIA CUDA, la bibliothèque CUBLAS [NVI08a], est aussi disponible. CUBLAS se compose des fonctions au niveau API (Application Programming Interface). Le modèle basique pour utiliser CUBLAS est de (i) créer des matrices et vecteurs dans la mémoire GPU, (ii) les remplir avec les données, (iii) appeler une suite de fonctions de CUBLAS et (iv) copier les résultats vers la mémoire CPU. Les fonctions de CUBLAS sont exécutées implicitement parallèlement sur le GPU.

Comme les calculs nécessaires pour l'AFC sont des manipulations de matrices, nous proposons une version parallèle de l'algorithme incrémental d'AFC qui utilise CUBLAS. L'implémentation incrémentale parallèle du calcul l'AFC (décrite dans l'algorithme 4) manipule des matrices sur le GPU pour accélérer des calculs.

Pour chaque étape incrémentale i, on charge un bloc de données $\mathbf{F}_{[i]}$ en mémoire CPU et le copie vers la mémoire GPU ; puis les formules 4.3 et 4.5 sont calculées de manière parallèle sur GPU (en appelant les fonctions correspondant de CUBLAS). Lorsque les itérations se terminent, on calcule $\mathbf{A}$ sur GPU avant de la recopier vers la mémoire CPU (lignes 9–10). L'algorithme calcule K vecteurs propres de la matrice $\mathbf{A}$ sur le CPU [2] et les copie vers la mémoire GPU pour l'étape de projection (lignes 11–12). Pour chaque itération dans l'étape de projection, on charge un bloc de données en mémoire CPU, la copie vers la mémoire GPU, calcule la formule de projection 4.8 et recopie le

2. Pour l'instant, cette étape est effectuée sur CPU car la décomposition en valeurs propres et vecteurs propres d'une matrice n'est pas implémentée sur GPU. On peut utiliser l'implémentation de la décomposition QR sur GPU [KCR09] pour accélérer la décomposition en valeurs propres et vecteurs propres. Il est également possible d'implémenter un algorithme itératif parallélisable comme l'algorithme de Jacobi sur GPU.

résultat vers la mémoire CPU (lignes 13–18).

Algorithme 4 : Algorithme incrémental parallèle pour calculer l'AFC

```
1  Q = 0
2  C = 0
3  pour i = 1 à B faire
4  |  charger le bloc F_[i] en mémoire de CPU
5  |  copier F_[i] vers mémoire de GPU
6  |  calculer P_[i], Q_[i] et C_[i] à partir de F_[i]
7  |  Q = Q + Q_[i]
8  |  C = C + C_[i]
9  A = CQ^-1
10 recopier A vers mémoire CPU
11 calculer K premiers vecteurs propres de A sur CPU
12 copier K vecteurs propres vers mémoire GPU
13 pour i = 1 à B faire
14 |  charger le bloc F_[i] en mémoire CPU
15 |  copier F_[i] vers mémoire de GPU
16 |  calculer P_[i] à partir de F_[i]
17 |  calculer la projection Z_[i] = P_[i]^-1 F_[i] Q^-1 U
18 |  recopier Z_[i] vers mémoire CPU
```

4.1.3 Parallélisation de la recherche d'image par AFC sur GPU

La méthode de recherche d'images approximative basée sur la qualité de représentation que nous avons proposée permet d'accélérer de 3 à 14 fois la recherche par rapport à une recherche exhaustive sans diminuer la qualité des résultats. Il s'agit d'une méthode en deux étapes : filtrage des images non pertinentes et raffinement des résultats sur une liste d'images candidates. Pour une base d'un million d'images, la plupart du temps de calcul est consacré à l'étape de filtrage (cf. Figure 4.1). Ceci nous motive à paralléliser cette étape afin d'accélérer encore la recherche. La parallélisation est réalisée sur le GPU en utilisant CUDA.

Représentation des fichiers inversés et parallélisation des calculs

L'étape de filtrage dans la recherche en deux étapes consiste à calculer les fréquences de chaque image dans la base et à rejeter les images non pertinentes pour la requête en se basant sur leur fréquence (cf. Section 3.5.3). Nous présentons maintenant la représentation des fichiers inversés et son utilisation pour paralléliser l'étape de filtrage sur le GPU.

Pour une parallélisation efficace des calculs sur le GPU, la dépendance des données entre *threads* doit être évitée, c'est à dire que chaque *thread* ne doit manipuler que ses

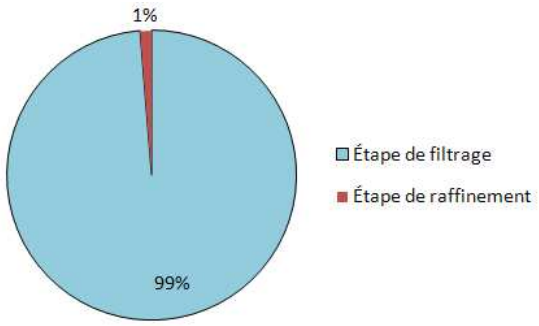


FIGURE 4.1 – Temps de calcul de l'étape de filtrage et de l'étape de raffinement dans la recherche sur une base d'un million d'images.

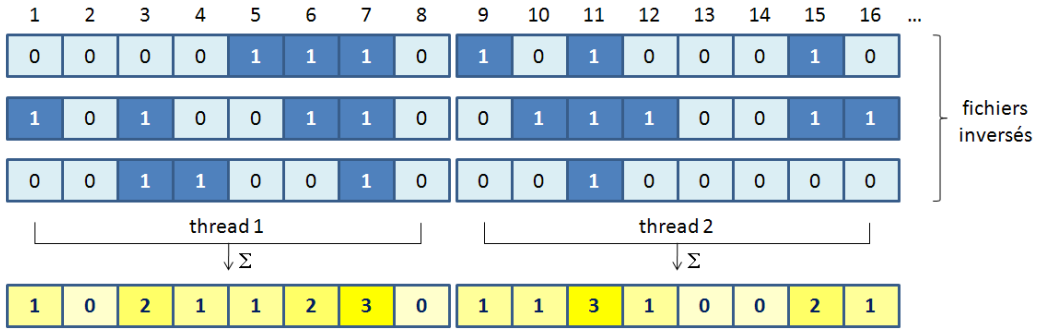


FIGURE 4.2 – Calcul parallèle de la fréquence des images sur GPU : chaque *thread* calcule la fréquence de 8 images.

propres données. C'est la raison pour laquelle nous représentons les fichiers comme des vecteurs de *bits*. Un fichier inversé représente la présence ou l'absence des images dans le fichier inversé par des *bits* 0 (absence) et 1 (présence). Soit $F_\alpha^\sigma, \sigma \in \{+,-\}$ est un fichier inversé, si l'image i appartient à ce fichier inversé, le $i^{ème}$ *bit* du vecteur de *bits* correspondant est mis à 1.

Une telle représentation permet de compresser les fichiers inversés pour qu'ils puissent tenir dans la mémoire GPU et de calculer la fréquence des images de manière indépendante dans chaque *thread*. Chaque *thread* calcule la fréquence de 8 images (un octet représente la présence ou l'absence de 8 images). La figure 4.2 illustre la parallélisation du calcul de la fréquence des images en utilisant les *threads* GPU.

Filtrage parallèle sur GPU

Avec la représentation des fichiers inversés proposée ci-dessus, l'étape de filtrage des images non pertinentes commence par le calcul parallèle des fréquences de chaque image de la base. Pour une image requête, on détermine d'abord les fichiers inversés correspondants et on transfère les indices de ces fichiers vers la mémoire GPU. Les

threads calculent indépendamment et parallèlement la fréquence des images en faisant une somme sur les bits correspondant des fichiers inversés (cf. Figure 4.2). Après avoir calculé les fréquences, on copie le résultat vers la mémoire CPU et on détermine le seuil θ comme dans le cas de la version non parallèle. On obtient ainsi la même liste d'images candidates.

4.1.4 Expérimentations

Nous avons réalisé les expérimentations sur la base Nistér-Stewénius (cf. Section 3.6 du chapitre 3). Pour évaluer la passage à l'échelle des méthodes proposées, nous avons fusionné la base Nistér-Stewénius avec 100 000, 200 000, 500 000 et un million d'images téléchargées à partir de FlickR [3]. Le nombre de mots visuels est fixé à 5 000.

Les algorithmes sont écrits en C++ utilisant la bibliothèque LAPACK [ABB+99] pour le calcul des valeurs propres et vecteurs propres. Nous avons utilisé la bibliothèque ATLAS [WPD01] pour optimiser les manipulations des matrices sur CPU. La version incrémentale parallèle de l'AFC sur GPU est implémentée en utilisant CUBLAS et l'algorithme de filtrage des images non pertinentes est parallélisé en utilisant CUDA sur une ou deux carte(s) graphique(s) NVidia GeForce GTX-280 avec 1GO de mémoire. Toutes les expérimentations sont réalisées sur une machine Intel Xeon bi-processeurs Quad-core, 3.2GHz, 8GO de mémoire vive. Après avoir effectué l'AFC sur les images, nous avons conservé les 500 premiers axes. La mesure du cosinus est utilisée pour calculer la similarité entre les images. La méthode de recherche approximative utilise les fichiers inversés basés sur la qualité de représentation. Le seuil $\hat{\theta}$ est choisi tel que la liste d'images candidates $\mathcal{H}_{\hat{\theta}}$ contienne au moins 500 images.

Le tableau 4.1 présente le temps d'exécution (en seconde) des deux algorithmes proposés sur des bases d'images avec différentes tailles. La version parallèle sur une carte GPU est environ 12 fois (pour la construction de la matrice $\mathbf{A}$ et la projection des images sur les axes factoriels) et 7 fois (au total) plus rapide que la version non parallèle car le calcul des valeurs propres / vecteurs propres est constant pour toutes les expérimentations ce qui est normal puisque le calcul se fait sur CPU. Avec deux cartes GPU le premier facteur est multiplié par 2 et le second par 1.5.

Le tableau 4.2 montre le temps de réponse (en millisecondes) et la précision aux 3 premières images retournées (P@3) pour différentes méthodes de recherche : la méthode approximative avec le filtrage des images non pertinentes (effectué sur GPU et sur CPU) et la méthode exhaustive. Sur une base d'un million d'images, la méthode avec filtrage des images non pertinentes effectué sur le GPU est environ 10 fois plus rapide que la méthode sans parallélisation du filtrage et de 114 fois plus rapide qu'une recherche exhaustive. Étonnamment, la recherche approximative donne de meilleurs résultats que la méthode exhaustive. Ceci est dû au fait que la qualité de représentation est un indicateur pertinent de l'AFC pour interpréter la proximité des images.

3. `http://www.flickr.com`

TABLE 4.1 – Comparaison du temps de calcul (en seconde) des deux algorithmes (parallèle et non parallèle) pour calculer l'AFC. (a) : temps de calcul pour l'étape de construction de la matrice **A** ; (b) : temps de calcul des valeurs propres/vecteurs propres ; (c) : temps de calcul pour l'étape de projection et Total = (a) + (b) + (c).

Taille de la base	CPU				
	(a) : Mat. **A**	(b) : Vec. prop.	(c) : Projection	(a) + (c)	Total
100K	314.21	63.6	31.47	345.68	409.28
200K	628.54	63.6	63.02	691.56	755.16
500K	1 569.02	63.6	157.40	1 726.42	1 790.02
1M	3 136.42	63.6	315.10	3 451.52	3 515.12

Taille de la base	1 GPU					Gain	
	(a)	(b)	(c)	(a) + (c)	Total	(a) + (c)	Total
100K	26.16	63.6	3.59	29.75	93.35	11.62	4.38
200K	51.91	63.6	7.14	59.05	122.65	11.71	6.16
500K	129.16	63.6	17.80	146.96	210.56	11.75	8.50
1M	257.91	63.6	35.57	293.48	357.08	11.76	9.84
					Moyenne	11.71	7.22

Taille de la base	2 GPU					Gain	
	(a)	(b)	(c)	(a) + (c)	Total	(a) + (c)	Total
100K	13.98	63.6	2.30	16.29	79.89	21.22	5.12
200K	26.88	63.6	4.08	30.96	94.56	22.34	7.99
500K	65.76	63.6	9.40	75.16	138.76	22.97	12.90
1M	130.46	63.6	18.21	148.67	212.27	23.22	16.56
					Moyenne	22.44	10.64

4.2 Combinaison de l'AFC avec d'autres méthodes

4.2.1 Mesure de dissimilarité contextuelle

La mesure de dissimilarité contextuelle, (MDC ou CDM : abréviation de Contextual Dissimilarity Measure) [JHS07] est une intégration des informations contextuelles à la recherche par la similarité. La mesure prend en compte la structure de voisinage des images pour corriger la *symétrie de voisinage* du schéma de recherche des k plus proches voisins (k-PPV). Nous associons à chaque image un poids inversement proportionnel à la densité de la région où se trouve cette image. Cette pondération favorise les images isolés et pénalise les images se trouvant dans les régions denses.

Considérons un voisinage $\mathcal{N}(i)$ d'une image i défini par les $n_{\mathcal{N}}$ plus proches voisins de i obtenus par un schéma de recherche des k-PPV ($k \equiv n_{\mathcal{N}}$). Nous définissons la distance de voisinage $d_{\mathcal{N}}(i)$ comme la moyenne des distances de l'image i à toutes les

TABLE 4.2 – Comparaison des méthodes de recherche.

Taille de la base	Filtrage sur GPU		Filtrage sur CPU		Recherche exhaustive	
	temps (ms)	P@3	temps (ms)	P@3	temps (ms)	P@3
100K	1.47	0.681	7.56	0.681	94.32	0.676
200K	2.09	0.665	13.53	0.665	179.76	0.660
500K	4.17	0.641	33.07	0.641	435.00	0.639
1M	7.53	0.625	79.99	0.625	860.88	0.623

images dans son voisinage $\mathcal{N}(i)$:

$$d_{\mathcal{N}}(i) = \frac{1}{n_{\mathcal{N}}} \sum_{j \in \mathcal{N}(i)} d(i,j) \tag{4.9}$$

où $d(i,j)$ est la distance (ou une mesure de dissimilarité quelconque) entre les images i et j. La quantité $d_{\mathcal{N}}(i)$ est calculée pour toutes les images de la base. La mesure de dissimilarité contextuelle $d_{MDC}(i,j)$ entre 2 images i et j est définie par :

$$d_{MDC}(i,j) = d(i,j) \left(\frac{\left(\bar{d}_{\mathcal{N}}\right)^2}{d_{\mathcal{N}}(i) d_{\mathcal{N}}(j)} \right)^{\beta} \tag{4.10}$$

où $0 < \beta < 1$ est le facteur de lissage et $\bar{d}_{\mathcal{N}}$ est la moyenne géométrique des distances de voisinage $r(i)$ obtenue par :

$$\bar{d}_{\mathcal{N}} = \prod_{i=1}^{M} \left(d_{\mathcal{N}}(i)\right)^{\frac{1}{M}} \tag{4.11}$$

Il est possible d'utiliser la moyenne algébrique au lieu de la moyenne géométrique et on obtient un résultat similaire.

La formule 4.10 peut se réécrire :

$$d_{MDC}(i,j) = d(i,j) \left(\frac{\left(\bar{d}_{\mathcal{N}}\right)^2}{d_{\mathcal{N}}(i) d_{\mathcal{N}}(j)} \right)^{\beta} \tag{4.12}$$

$$= d(i,j) \left(\frac{\bar{d}_{\mathcal{N}}}{d_{\mathcal{N}}(i)} \right)^{\beta} \left(\frac{\bar{d}_{\mathcal{N}}}{d_{\mathcal{N}}(j)} \right)^{\beta} \tag{4.13}$$

$$= d(i,j)\delta(i)\delta(j) \tag{4.14}$$

où

$$\delta(i) = \left(\frac{\bar{d}_{\mathcal{N}}}{d_{\mathcal{N}}(i)} \right)^{\beta} \tag{4.15}$$

Les k plus proches voisins d'une requête $\mathbf{r}$ sont obtenus par :

$$k\text{-PPV}(\mathbf{r}) = k\text{-}\operatorname*{argmin}_{i} \{d(i,\mathbf{r})\delta(i)\} \tag{4.16}$$

Les $\delta(i)$ sont appelés *termes de régularisation* et stockés dans la base.

4.2.2 Intégration de la MDC dans la recherche approximative par AFC

Quand la taille de la base d'images est grande, un inconvénient de la MDC est que le calcul des termes de régularisation implique le calcul des distances pour chaque paire d'images de la base. La complexité du calcul est ainsi quadratique en fonction du nombre d'images. Pour réduire la complexité, [JHS07] a proposé une méthode de recherche approximative en regroupant les images en *clusters*. Les plus proches voisins d'une image i sont cherchés dans un nombre limité (par ex., 3, soit 1% de la base) de clusters les plus proches de i. Le choix des clusters plus proches d'une image se base sur la distance de cette image aux centres des clusters. Le nombre de clusters voisins d'un cluster donné augmente de manière exponentielle avec le nombre de dimensions. Donc, si le nombre de clusters les plus proches est trop petit, le risque devient grand de ne pas obtenir un bon voisinage et la précision est dégradée. Par ailleurs, les méthodes basées sur le clustering dégradent la qualité des résultats lorsque les données ne sont pas bien organisées en agrégats (cf. Section 2.4.3).

C'est pour cela que nous nous proposons d'utiliser notre méthode de recherche approximative se basant sur les fichiers inversés pour calculer les termes de régularisation $\delta(i)$. Cette méthode de recherche approximative permet de trouver de bons voisins d'une requête en examinant seulement un tout petit ensemble d'images (par ex., 0.05% de la taille de la base). Ainsi la complexité du calcul est diminuée et la précision est améliorée. Nous exploitons cette mesure de dissimilarité contextuelle de deux manières :

- hors ligne : nous calculons les termes de régularisation $\delta(i)$ une seule fois avant la recherche. Cette solution est appropriée pour les bases d'images statiques où la mise à jour est rare.
- à la volée : au cours de la recherche, nous calculons les termes de régularisation $\delta(i)$ pour des images dans la liste d'images candidates seulement. De cette manière, le problème de la mise à jour n'existe plus.

4.2.3 Forêt aléatoire

Proposé par L. Breiman, la forêt aléatoire (Random Forest) [Bre01] est un outil pour la classification supervisée, la régression et le clustering des données. L'algorithme de forêts aléatoires crée un ensemble d'arbres de décision afin de réduire l'erreur de biais et d'obtenir une faible corrélation entre les arbres.

Considérons une tâche de classification supervisée de M individus $\mathbf{x} \in \mathbf{X}$ à N attributs. Un arbre de décision (noté DT) de la forêt de T arbres (noté $\mathrm{RF} = \{\mathrm{DT}_j\}_{j=1..T}$) est créé de la façon suivante :

- tirage avec remise depuis l'ensemble d'apprentissage $\mathbf{X}$ d'un échantillon *bootstrap* (noté $\mathbf{X}^{(i)}, i \in [1;T]$) qui est utilisé pour la construction de l'arbre ;
- recherche d'une meilleure coupe pour chaque nœud de décision à partir d'un sous-ensemble aléatoire de N_0 attributs ($N_0 < N$, par ex., $N_0 = \sqrt{N}$) ;
- construction de l'arbre le plus profond possible (sans élagage).

La recherche de la meilleure coupe se base sur un sous ensemble d'attributs tiré

aléatoirement et sur une mesure d'impureté telle l'entropie de Shannon [Sha01] utilisée dans C4.5 [Qui93] ou l'indice de Gini (*Gini index*) utilisé dans CART [BFSO84].

Pour prédire l'étiquette d'un nouvel individu, on utilise un vote majoritaire des arbres de la forêt dans le cas de la classification ou la moyenne des prédictions des arbres dans le cas de la régression. L'algorithme de forêts aléatoires donne de bons résultats. Par ailleurs, il est rapide et robuste aux données bruitées.

Breiman a étendu les forêts aléatoires au cas de l'apprentissage non supervisé [Bre01]. Étant donné un ensemble de M individus non étiquetés, on génère un autre ensemble de données à partir de l'ensemble original et on étiquette l'ensemble original par la classe $+1$ et l'ensemble synthétique par la classe -1. La génération de l'ensemble de données synthétiques se base sur la distribution de l'ensemble original. En effet, on génère M individus synthétiques. Pour chacun, la valeur du $j^{ème}$ attribut est tirée aléatoirement de M valeurs du $j^{ème}$ attribut des individus originaux :

$$\mathbf{y}(j) \quad = \quad \underset{\mathbf{x}\in\mathbf{X}}{rand}\{\mathbf{x}(j)\} \tag{4.17}$$

où $\mathbf{y}(j)$ est la valeur du $j^{ème}$ attribut d'un individu synthétique $\mathbf{y}$; $\mathbf{x}(j)$ est la valeur du $j^{ème}$ attribut de l'individu $\mathbf{x} \in \mathbf{X}$; et *rand* est une fonction qui tire aléatoirement une valeur à partir d'un ensemble de valeurs.

Après avoir obtenu deux ensembles étiquetés ($2M$ individus), on construit la forêt aléatoire à partir de ces deux ensembles comme dans le cas de l'apprentissage supervisé présenté ci-dessus. On définit ensuite la proximité entre deux individus.

Définition 4.1 (Proximité selon une forêt aléatoire) – *La proximité selon une forêt aléatoire entre deux individus i et j, notée* $\text{prox}(i,j)$ *est le nombre de feuilles d'un des arbres de la forêt où les deux individus i et j se trouvent :*

$$\text{prox}(i,j) \quad = \quad \big|\{\mathbf{f} \mid i \in \mathbf{f} \text{ et } j \in \mathbf{f}\}\big| \tag{4.18}$$

où $\mathbf{f}$ est une feuille d'un arbre de la forêt aléatoire.

Après avoir défini la proximité entre des individus, on peut effectuer des tâches d'analyse comme le clustering ou la réduction de dimensions.

4.2.4 Arbres obliques

La construction d'un arbre de décision dans les algorithmes de forêt aléatoire n'utilise qu'un seul attribut parmi un sous-ensemble d'attributs tiré aléatoirement pour séparer les individus à chaque niveau de l'arbre [BFSO84, Qui93]. Beaucoup d'information peut être perdue en cas de dépendances entre attributs. L'exemple de la figure 4.3 montre qu'il n'existe aucune coupe perpendiculaire aux axes permettant de séparer totalement les individus en une seule fois. Mais la coupe oblique H1 (combinaison de deux attributs) classifie parfaitement les individus en deux classes. Pour remédier à ce problème, on peut construire des arbres multivariés qui effectuent des coupes obliques pour séparer les individus. La complexité de la construction d'un arbre oblique est NP-difficile [Hea92].

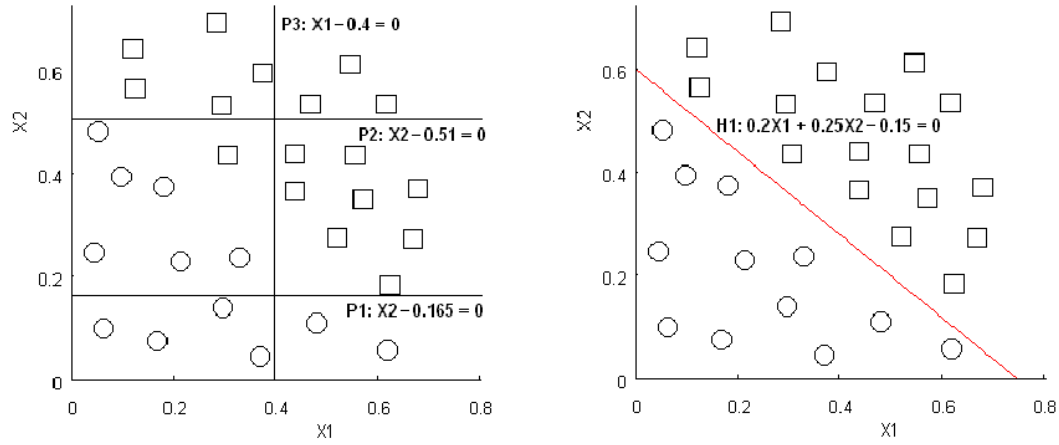


FIGURE 4.3 – Coupe selon un seul attribut (à gauche) et selon deux atributs (à droite).

Breiman *et al.* ont aussi proposé dans [BFSO84] une méthode de construction d'arbres obliques qui combine linéairement les attributs en utilisant des coefficients générés aléatoirement dans l'intervalle $[-1; 1]$.

Cependant cette approche ne peut traiter qu'un petit nombre d'attributs à cause de l'explosion combinatoire. Murthy *et al.* ont proposé OC1 [MKSB93], un système d'induction d'arbres obliques qui fait une recherche locale (méthode heuristique) pour trouver une bonne coupe oblique (hyperplan). Dans [WBCST99], Wu *et al.* ont étendu l'algorithme OC1 en modifiant profondément la recherche d'une coupe oblique. Ils ont proposé d'utiliser les SVMs (support vector machines) pour trouver des hyperplans optimaux plus robustes que l'algorithme OC1. Cette approche améliore la performance de l'algorithme OC1. Cependant, l'utilisation d'un SVM standard se ramène à résoudre un programme quadratique dont la complexité est au moins égal au carré du nombre d'individus.

Dans [DLPP09], nous avons proposé un algorithme de forêt aléatoire (noté RF-ODT) qui construit un ensemble d'arbres obliques de la même manière que celui proposé par Breiman. La seule différence est qu'un arbre oblique dans la forêt aléatoire utilise un hyperplan général au lieu d'un hyperplan perpendiculaire aux axes pour effectuer les coupes obliques. La recherche des hyperplans obliques se fonde sur l'algorithme PSVM (proximal support vector machine) [FM01] car l'algorithme PSVM est très rapide par rapport aux algorithmes SVM standard. Le PSVM résout un système d'équations linéaires au lieu d'une programmation quadratique comme c'est le cas pour un SVM standard. La complexité du PSVM est donc linéaire en fonction du nombre d'individus et il est capable de traiter de grands ensembles de données. L'évaluation de RF-ODT dans le domaine de la classification supervisée montre que RF-ODT donne de meilleurs résultats que d'autres méthodes d'apprentissage automatique comme le LibSVM [CL01], C4.5 [Qui93], et la forêt aléatoire d'arbres de décision C4.5.

4.2.5 Recherche d'images par forêt aléatoire

Nous cherchons à intégrer la proximité selon une forêt aléatoire dans la recherche d'images à l'aide de l'AFC. Après avoir fait l'AFC des images, nous avons une représentation réduite des images sur les axes factoriels. Nous construisons ensuite une forêt aléatoire avec cette nouvelle représentation des images.

La recherche d'images par forêt aléatoire se compose de deux étapes.

1. **Indexation** – Cette étape se fait hors ligne en plaçant les images de la base dans les feuilles correspondantes (en appliquant les règles de déduction SI – ALORS).
2. **Recherche** – L'étape de recherche proprement dite consiste d'abord à déterminer les feuilles, pour chaque arbre de décision dans la forêt, où l'on va mettre la requête. La proximité de toutes les images se trouvant dans une feuille où l'on a mis la requête est alors augmentée de 1.

Nous avons exploité la proximité selon une forêt aléatoire de deux manières.

- On utilise directement cette proximité comme une mesure de similarité. Dans ce cas, on remplace la similarité du cosinus par la proximité selon une forêt aléatoire.
- La proximité n'est utilisée que dans l'étape de filtrage pour filtrer les images non pertinentes (cf. la méthode de recherche en deux étapes). Dans ce cas, on remplace les fichiers inversés par les feuilles des arbres de la forêt. La proximité selon une forêt aléatoire joue le même rôle que la fréquence des images. Une recherche séquentielle dans la liste d'images candidates est nécessaire pour raffiner les résultats.

La complexité en temps de l'algorithme de recherche par forêt aléatoire est théoriquement $O(T \log M)$ où T est le nombre d'arbres de la forêt et M est le nombre d'individus de la base. Par ailleurs, la détermination des feuilles pour une requête dans une forêt aléatoire est indépendante pour chaque arbre. Il est possible de répartir les arbres pour paralléliser la recherche.

4.2.6 Expérimentations

Nous évaluons la combinaison de l'AFC avec la mesure de dissimilarité contextuelle et la forêt aléatoire. Les expérimentations sont effectuées sur la base Nistér-Stewénius fusionnée avec des images de FlickR.

Mesure de similarité contextuelle

Dans le tableau 4.3, nous comparons la combinaison entre notre algorithme de recherche approximative basée sur la qualité de représentation et la MDC (noté AFC-MDC(1) : combinaison hors ligne et AFC-MDC(2) : combinaison à la volée) à la méthode proposée dans [JHS07] qui combine le *clustering* des images (sur la représentation par des mots visuels) et la MDC (noté Clustering-MDC). Pour la méthode Clustering-MDC, on regroupe les images en 500 clusters et prend 50 clusters (soit 10% de la base) les plus proches à la requête pour chercher des images similaires. Le paramètre $n_{\mathcal{N}}$ pour calculer les termes de régularisation $\delta(i)$ est choisi égal à 10 et nous avons fixé β égal à

TABLE 4.3 – Comparaison des méthodes de recherche utilisant MDC.

Bases d'images	AFC-MDC(1)		AFC-MDC(2)		Clustering-MDC	
	temps (ms)	P@3	temps (ms)	P@3	temps (ms)	P@3
100K	**7.9**	0.741	35.8	**0.749**	17.2	0.696
200K	**12.8**	0.725	40.6	**0.735**	30.5	0.683
500K	**32.8**	0.705	60.7	**0.715**	78.6	0.660
1M	**86.4**	0.688	115.6	**0.701**	171.3	0.643

0.6 comme dans [JHS07]. Le seuil $\hat{\theta}$ pour les méthodes basées sur les fichiers inversés de l'AFC est choisi tel que la liste d'images candidates $\mathcal{H}_{\hat{\theta}}$ contiene au moins 500 images.

Pour toutes les méthodes, nous calculons la *précision* aux 3 premières images retournées (P@3) comme mesure d'évaluation. Nous présentons également le temps de réponse de toutes les méthodes.

Les résultats montrent que l'intégration de la mesure de dissimilarité contextuelle améliore considérablement la pertinence des images retournées par rapport à la similarité du cosinus. Par ailleurs, notre méthode de recherche approximative basée sur les fichiers inversés fait mieux que la méthode basée sur le clustering en termes de qualité des résultats ainsi que le temps de réponse. L'intégration de MDC à la volée améliore légèrement la qualité des résultats par rapport à l'intégration hors ligne en ajoutant un coût supplémentaire pour calculer, à la volée, les termes de régularisation des images dans la liste d'images candidates.

Forêt aléatoire

Dans ces expérimentations, nous évaluons l'utilisation de la proximité selon une forêt aléatoire dans la recherche d'images par AFC. Les tests sont réalisés sur les bases : Caltech-4, Caltech-101 et Nistér-Stewénius. Nous avons développé notre algorithme de forêt aléatoire des arbres obliques (RF-ODT) en C++. Pour évaluer la performance des méthodes, nous calculons la *précision* aux 10, 20, 50, et 100 premières images retournés (pour les bases Caltech-4 et Caltech-101) et aux 3, 10 et 20 première images (pour la base Nistér-Stewénius).

Deux tableaux 4.4 et 4.5 présentent les résultats de l'utilisation de la proximité selon une forêt aléatoire pour la recherche d'images. Dans ces tableaux, K est le nombre d'axes conservés et T est le nombre d'arbres dans la forêt aléatoire. La méthode RF-ODT(1) utilise la proximité selon une forêt aléatoire comme mesure de similarité tandis que la méthode RF-ODT(2) l'utilise pour filtrer des images non pertinentes comme la méthode de recherche basée sur les fichiers inversés. Dans ce cas, les feuilles jouent le même rôle que les fichiers inversés. Une recherche séquentielle dans la liste d'images candidates (avec la similarité du cosinus) est nécessaire pour raffiner les résultats. Nous présentons aussi les résultats d'une recherche séquentielle avec la similarité cosinus (AFC + Cosinus) comme méthode de référence.

La première remarque est que plus T est grand, meilleurs sont les résultats quelle que soit l'utilisation de la proximité selon une forêt aléatoire. L'utilisation directe de

TABLE 4.4 – Combinaison de la proximité par forêt aléatoire avec l'AFC sur les bases Catech-4 et Caltech-101

Bases/Méthodes ↓		P@10	P@20	P@50	P@100	temps (ms)
Caltech-4 $K = 30$	RF-ODT(1), $T = 50$	0.964	0.958	0.945	0.928	**0.059**
	RF-ODT(1), $T = 100$	0.969	0.963	0.952	0.942	0.115
	RF-ODT(1), $T = 200$	0.970	0.964	0.954	0.945	0.276
	RF-ODT(1), $T = 500$	**0.972**	**0.966**	**0.956**	**0.948**	1.076
	RF-ODT(2), $T = 30$	0.971	0.965	0.953	0.941	0.081
	RF-ODT(2), $T = 50$	0.971	0.965	0.954	0.944	0.115
	RF-ODT(2), $T = 100$	0.971	0.964	0.954	0.945	0.169
	AFC + Cosinus	0.971	0.965	0.954	0.945	0.531
Caltech-101 $K = 100$	RF-ODT(1), $T = 100$	0.340	0.318	0.284	0.254	0.263
	RF-ODT(1), $T = 200$	0.349	0.326	0.291	0.262	0.604
	RF-ODT(1), $T = 500$	0.358	0.333	0.297	0.267	2.039
	RF-ODT(1), $T = 1000$	0.360	0.337	0.300	**0.270**	4.696
	RF-ODT(2), $T = 50$	0.369	0.342	0.299	0.262	0.331
	RF-ODT(2), $T = 100$	**0.371**	**0.344**	0.301	0.266	0.495
	RF-ODT(2), $T = 200$	**0.371**	**0.344**	**0.302**	0.267	0.865
	AFC + Cosinus	**0.371**	**0.344**	**0.302**	0.266	4.688

la proximité selon une forêt aléatoire ne donne de bons résultats que dans la base de Caltech-4. Pour les bases Caltech-101 et Nistér-Stewénius, elle est beaucoup moins bonne que la similarité du cosinus. Par contre, quand la proximité par forêt aléatoire est utilisée pour filtrer les images non pertinentes, nous obtenons d'aussi bons résultats qu'avec une recherche séquentielle avec un temps de réponse plus petit (4 à 10 fois plus rapide).

4.3 Synthèse

Nous avons présenté, dans ce chapitre, le passage à l'échelle de l'AFC et la combinaison de l'AFC avec la mesure de dissimilarité contextuelle, d'une part et une forêt aléatoire d'autre part pour la recherche d'images par le contenu. Pour traiter de grand tableaux de contingence, nous avons proposé un algorithme incrémental pour calculer l'AFC. L'algorithme découpe le tableau en blocs par lignes et les charge successivement en mémoire. De cette manière, il n'a besoin de charger le tableau de contingence que deux fois (une pour calculer la matrice à diagonaliser et l'autre pour calculer la projection). Nous avons également parallélisé l'algorithme incrémental sur GPU pour pouvoir traiter rapidement de grands ensembles de données. Bien que notre méthode de recherche basée sur les fichiers inversés accélère beaucoup la recherche exhaustive (de 4 à 20 fois plus rapide), nous avons essayé d'accélérer encore la recherche en parallélisant l'étape de filtrage de notre méthode sur GPU. La méthode de recherche parallèle sur GPU est 10 fois plus rapide que la version non parallèle sur CPU et environ 100 fois plus rapide qu'une recherche séquentielle sur CPU sans perdre de qualité sur les

TABLE 4.5 – Combinaison de la proximité par forêt aléatoire avec l'AFC sur la base Nistér-Stewénius avec $K = 500$

Méthodes ↓	P@3	P@10	P@20	temps (ms)
RF-ODT(1), $T = 100$	0.547	0.196	0.105	**0.403**
RF-ODT(1), $T = 200$	0.613	0.213	0.113	0.922
RF-ODT(1), $T = 500$	0.667	0.227	0.120	3.256
RF-ODT(1), $T = 1000$	0.687	0.233	0.122	6.817
RF-ODT(1), $T = 2000$	0.699	0.236	0.123	13.587
RF-ODT(2), $T = 100$	0.731	0.242	0.125	1.398
RF-ODT(2), $T = 200$	0.739	0.246	0.128	2.131
RF-ODT(2), $T = 300$	0.741	0.246	**0.129**	2.827
RF-ODT(2), $T = 500$	**0.742**	**0.248**	**0.129**	4.259
AFC + Cosinus	**0.742**	**0.248**	**0.129**	16.449

résultats. L'intégration de la mesure de dissimilarité contextuelle à notre méthode de recherche améliore considérablement la pertinence des images retournées. Originellement, les termes de régularisation sont calculés hors ligne. Nous avons étendu notre méthode de recherche pour pouvoir calculer ces termes au cours de la recherche. Les tests numériques réalisés montrent que l'intégration de la MDC à la volée améliore la qualité de la recherche par rapport à la méthode hors ligne en ajoutant un coût supplémentaire. Bien que les résultats de l'utilisation directe de la proximité selon une forêt aléatoire à la place des mesures de similarité classiques comme L_1, L_2 ou similarité du cosinus ne soient pas très bons en général, l'utilisation de cette proximité pour une recherche en deux étapes, permet d'obtenir d'aussi bons résultats que dans le cas d'une recherche séquentielle en un temps de réponse court. Puisque la complexité du calcul des proximités par forêt aléatoire est sous linéaire avec le nombre d'images de la base ($O(T \log M)$), la recherche par forêt aléatoire peut être une méthode prometteuse dans l'avenir.

Chapitre 5

Visualisation des résultats de l'Analyse factorielle des correspondances sur les images

5.1 Introduction

La fouille de données [FGS96] vise à extraire des connaissances utiles cachées à partir de grandes bases de données. Cette activité est très liée à l'objectif des utilisateurs : c'est l'utilisateur qui peut déterminer si les résultats de la fouille sont pertinents pour lui ou non. Il est donc souhaitable que les outils de la fouille de données soient interactifs et permettent la participation de l'utilisateur. L'idée ici est d'augmenter l'interactivité au travers des techniques de visualisation. Il existe un grand nombre de méthodes de visualisation développées au cours de la dernière décennie dans différents domaines et qui ont été utilisées dans l'exploration des données et le processus d'extraction des connaissances [FGW01, Kei02]

Les méthodes de visualisation sont utilisées pour la sélection des données (phase de pré-traitement) et la visualisation des résultats (phase de post-traitement). Certaines méthodes récentes d'extraction visuelle de données [AEK00, DP04, Pou04] essaient de faire participer plus intensivement l'utilisateur dans le processus de fouille de données par l'utilisation de la visualisation. Cette coopération peut apporter des avantages comme l'utilisation de connaissances du domaine au cours de la construction du modèle, l'amélioration de la confiance et de la compréhensibilité des modèles obtenus ou l'utilisation des capacités humaines de reconnaissance des formes dans la phase de construction et exploration du modèle [Pou04].

En analyse de données textuelles, l'outil Bi-Qnomis [KBC06] développé par M. Kerbaol permet de visualiser les résultats et de trouver des thèmes pertinents dans un corpus textuel traité par l'analyse factorielle des correspondances (AFC). Les nuages de points (documents et mots) sont projetés sur un plan factoriel. Puis, pour chaque axe, on fait une liste des mots ayant une forte contribution (correspondant à une métaclé). Ces mots sont affichés à gauche et le titre des documents bien représentés sur ce plan

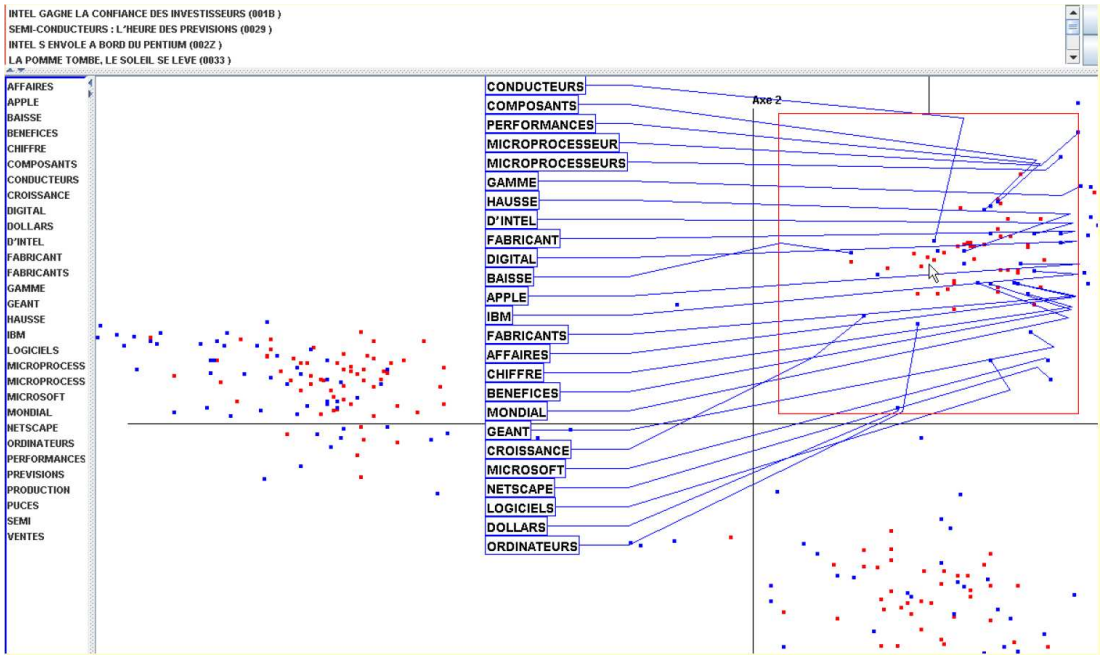


FIGURE 5.1 – Visualisation des documents et des mots avec l'outil Bi-Qnomis.

sont affichés en haut (Figure 5.1). L'utilisateur peut cliquer sur un titre pour voir en détail un document. En examinant les métaclés et les documents, un expert aura une vue synthétique du contenu de ces documents. Bi-Qnomis fournit aussi une visualisation des métaclés et leurs mots associés en utilisant un arbre hyperbolique (Figure 5.2).

Dans ce chapitre nous présentons un outil de visualisation des résultats d'AFC adaptée au cas des images. Comme il n'y a pas de « vrais » mots au sens littéral dans les images, on a utilisé des « mots visuels » à la place des mots textuels et les images sont considérées comme documents. L'outil graphique interactif que nous avons développé est appelé CAViz. CAViz permet aussi la découverte de thèmes dans des bases d'images. Les expérimentations sont réalisées sur la base Caltech-4 (cf. Section 3.6).

5.2 Projection sur le plan factoriel

Après avoir fait une AFC sur les images, on les projette sur les plans factoriels en s'aidant des indicateurs usuels de l'AFC : contribution à l'inertie et la qualité de représentation. La figure 5.3 présente la projection des images de la base Caltech-4 sur le plan factoriel 1–2 (c-à-d. le plan constitué des axes 1 et 2).

L'écran est divisé en deux parties : à gauche, nous projetons les nuages de points (images et/ou mots visuels) et la partie droite est réservée à l'affichage des images sélectionnées. Un point-image s'affiche en rouge, un point-mot visuel est de couleur bleue. L'utilisateur peut sélectionner une image ou un groupe d'images en pointant sur l'image intéressante. Toutes les images se trouvant dans un voisinage de rayon r de

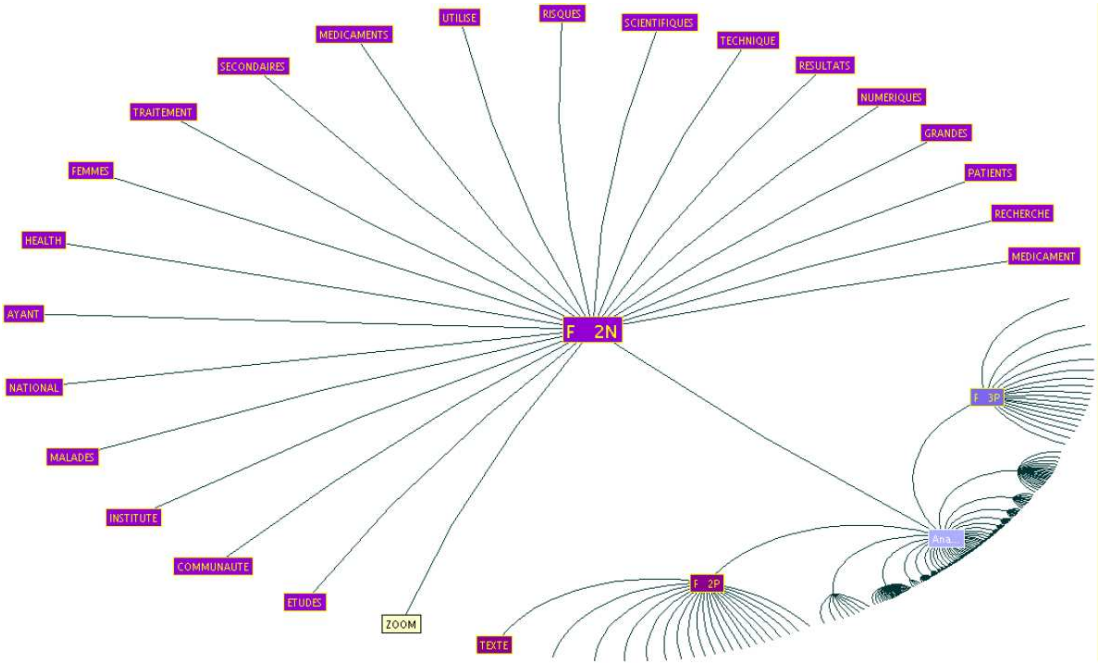


FIGURE 5.2 – Visualisation des métaclés par un arbre hyperbolique.

l'image intéressante sont également affichées à droite. Ceci nous donne tout de suite une vue générale du contenu de ces images. La couleur des points correspondant aux images sélectionnées sera modifiée de rouge en vert. Les mots visuels (sous forme d'une ellipse) seront superposés sur les images sélectionnées (voir la figure 5.4).

Pour se concentrer sur des images et/ou des mots intéressants, nous n'affichons, dans le plan, que les images et/ou les mots ayant une forte contribution, généralement 2 ou 3 fois la contribution moyenne. L'inertie totale sur un axe est égale à la valeur propre associée à cet axe. Le seuil est donc facile à déterminer. M. Kerbaol appelle métaclés les groupes des mots dont la contribution est très élevée sur un axe. Nous avons donc 2 métaclés par axe, une positive et une négative. Les mots visuels appartenant à chaque métaclé seront affichés sur l'image correspondante. Dans la figure 5.5, nous affichons les métaclés et leurs mots visuels. Les images se trouvant à chaque extrémité des axes correspondent à des métaclés. L'image en haut à gauche est superposée aux mots visuels proches d'elle. Il est facile de vérifier que la plupart de ces mots se trouvent à la fois dans la métaclé à gauche et la métaclé en bas.

L'information concernant une image est également extraite de façon interactive en sélectionnant l'image en question. Il y a 2 indicateurs pertinents pour interpréter le résultat de l'AFC : la qualité de représentation d'une part et la contribution à l'inertie d'autre part. Ces informations nous aideront à certaines tâches ultérieures. La figure 5.6 montre un exemple de l'extraction interactive de ces informations. L'image dans cet exemple est bien représentée par les axes 2, 3, 25, et 33 et contribue beaucoup aux axes 2, 25, 33 et 43.

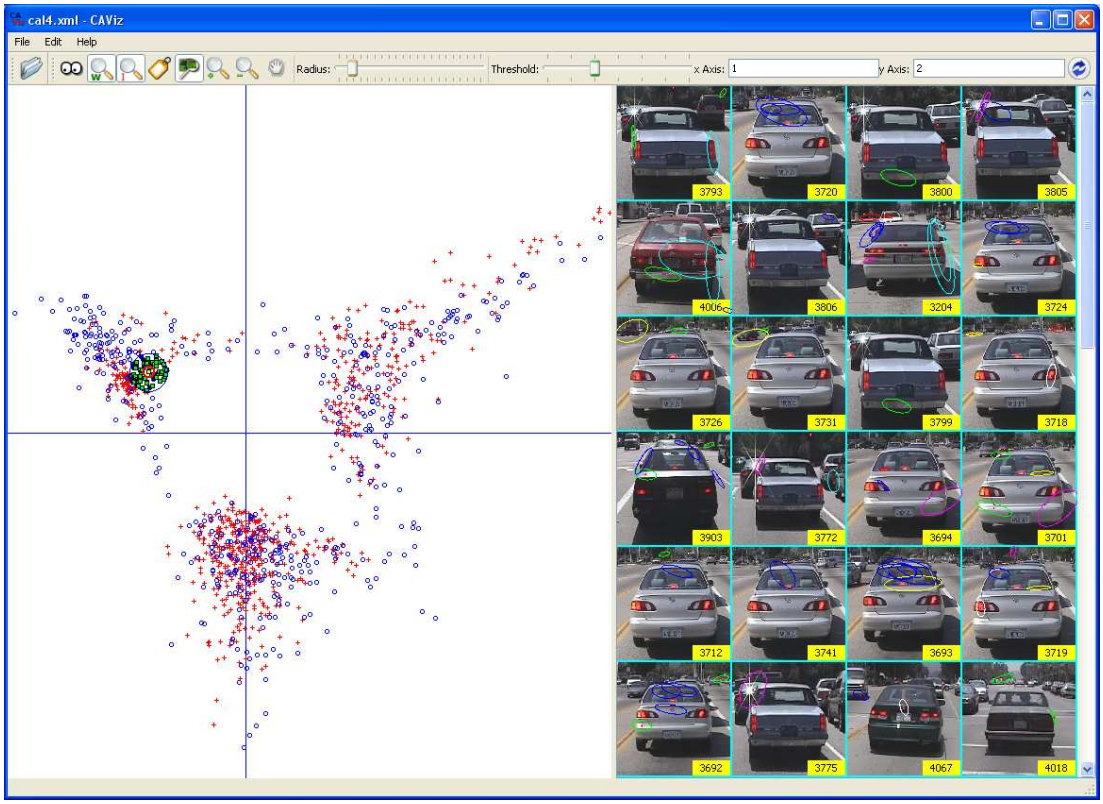


FIGURE 5.3 – Projection des images et des mots visuels sur le plan factoriel 1–2.

Un des avantages de l'AFC est que l'on peut afficher des mots visuels et des images dans un même plan. Les mots proches d'une image caractérisent bien cette image. Il est aisé de visualiser les mots caractérisant bien une ou un groupe d'images (correspondant à un thème) en sélectionnant l'image et affichant les mots proches de cette image. La figure 5.7 montre les mots qui représentent bien le thème « visage ».

5.3 Découverte des thèmes d'images

Après avoir affiché les nuages des points sur un plan factoriel, un groupe de points proches les uns des autres peut définir un thème dans ce plan factoriel. Néanmoins, pour pouvoir découvrir des thèmes plus fins, il faut chercher les axes (le plan) qui représentent ces thèmes. CAViz permet d'afficher les informations sur la qualité de représentation des images sur les axes et fournit un couplage des vues qui permet de voir la qualité de représentation d'un groupe d'images.

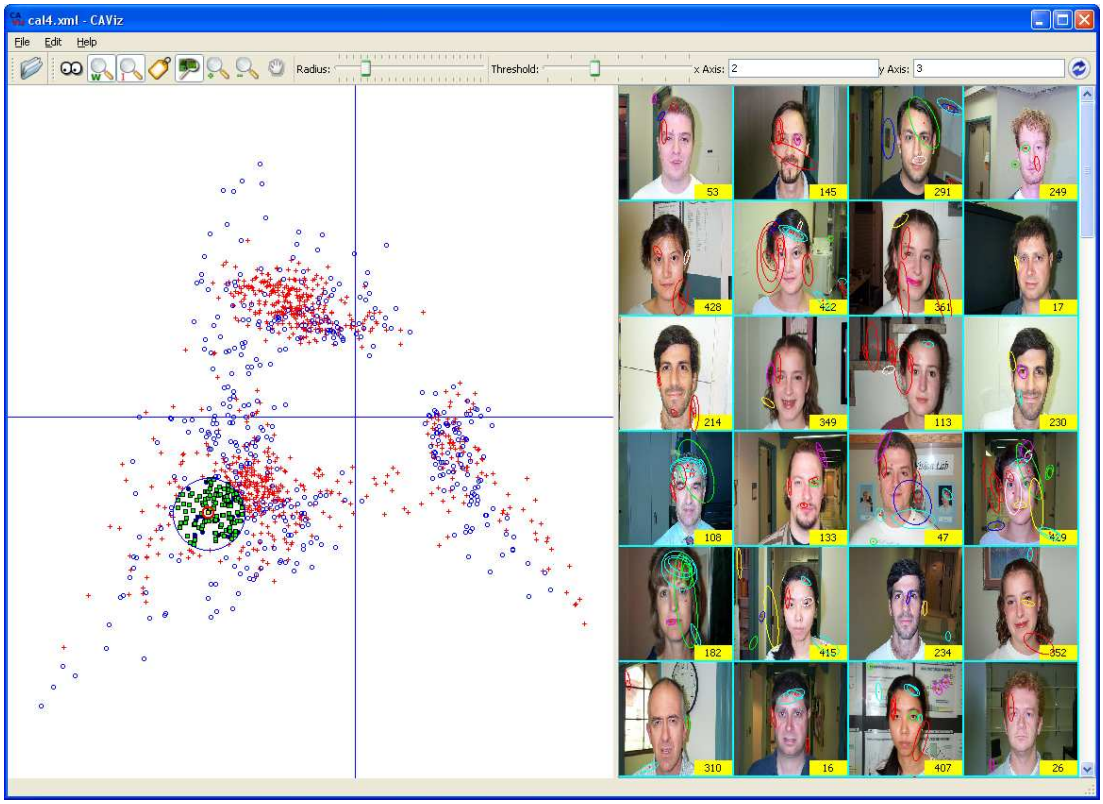


FIGURE 5.4 – Visualisation simultanée des images et des mots.

5.3.1 Qualité de représentation des images

Rappelons que la qualité de représentation d'un point i (correspondant à l'image i) sur l'axe α est le cosinus carré de l'angle entre l'axe α et le vecteur joignant le centre du nuage au point i. Si le cosinus carré d'une image sur un axe est grand, cet axe la représente bien. On utilise ce critère pour chercher les axes représentant bien les images et les mots. Les nuages de points sont projetés sur le premier plan factoriel (par ex. les axes 1 et 2). On extrait la qualité de représentation des images sur les axes en regardant les premiers facteurs dans l'histogramme de qualité de représentation. Ces informations nous donnent un guide pour chercher le bon plan factoriel qui représente bien un groupe d'images.

5.3.2 Couplage des vues

CAViz affiche les nuages de points dans la partie gauche et les images sélectionnées dans la partie droite. En cliquant une image à droite, le point correspondant à gauche est sélectionné et ses informations sont aussi affichées. Ce couplage des vues nous permet de sélectionner facilement les images intéressantes. La perception visuelle humaine est un bon outil pour la reconnaissance de formes. Nous sélectionnons les images similaires

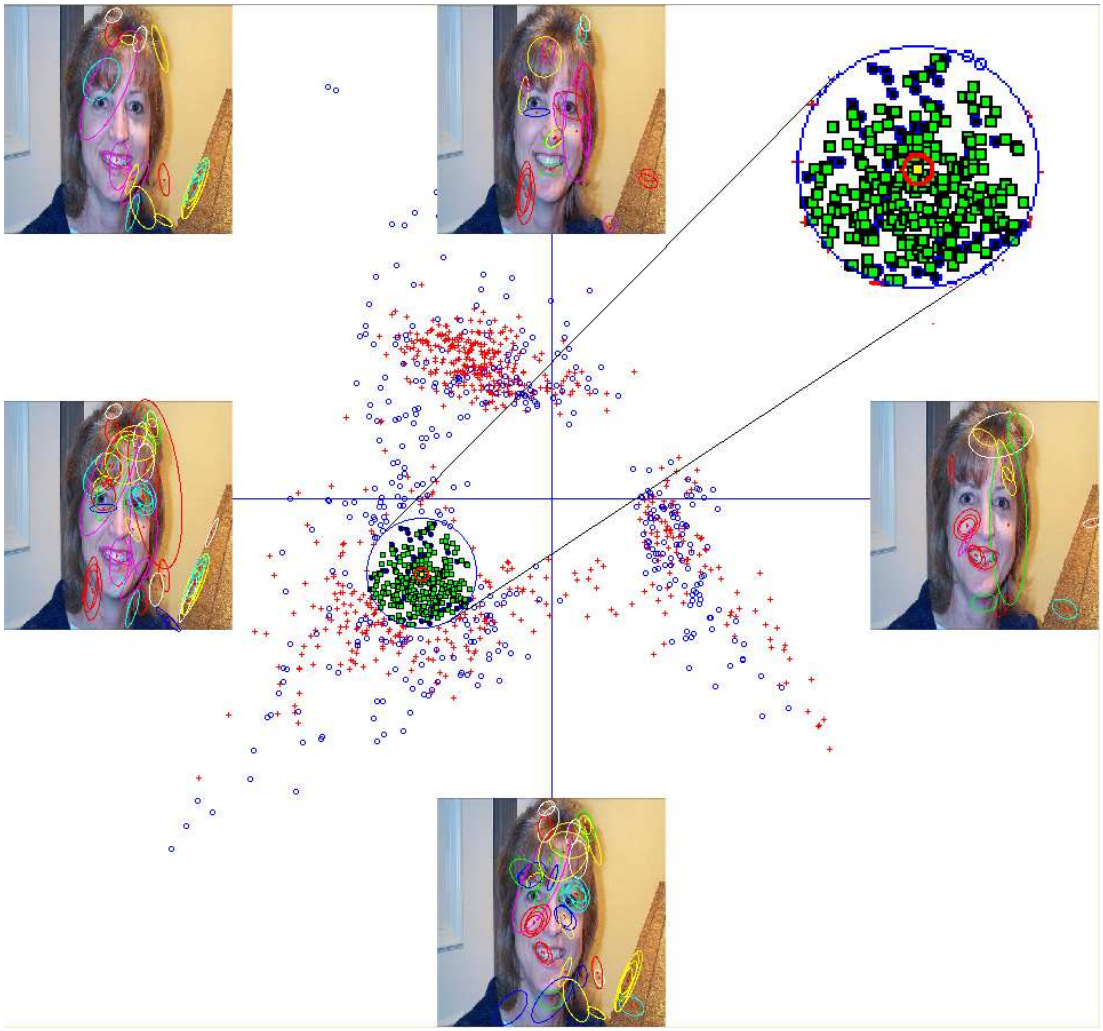


FIGURE 5.5 – Visualisation des métaclés

à droite et regardons la qualité de représentation de ces images. Si on trouve que ce sont les mêmes axes qui représentent bien les images sélectionnées à droite, on prend alors ces axes pour projeter les points.

5.3.3 Découverte de thèmes

Nous donnons ici une étude de cas sur la découverte de thèmes dans la base Caltech-4. Cette base contient 4090 images réparties en 5 catégories. On applique une AFC et on projette les points dans le premier plan et on pointe sur un groupe à gauche (correspondant aux voitures). Les images sont donc affichées à droite. On commence à sélectionner des images à droite en cliquant et regardant la qualité de représentation. On constate par exemple qu'il y a des images bien représentées par les axes 6 (négatif) et 7 (positif) et d'autres images bien représentées par les axes 11 (négatif) et 12 (positif)

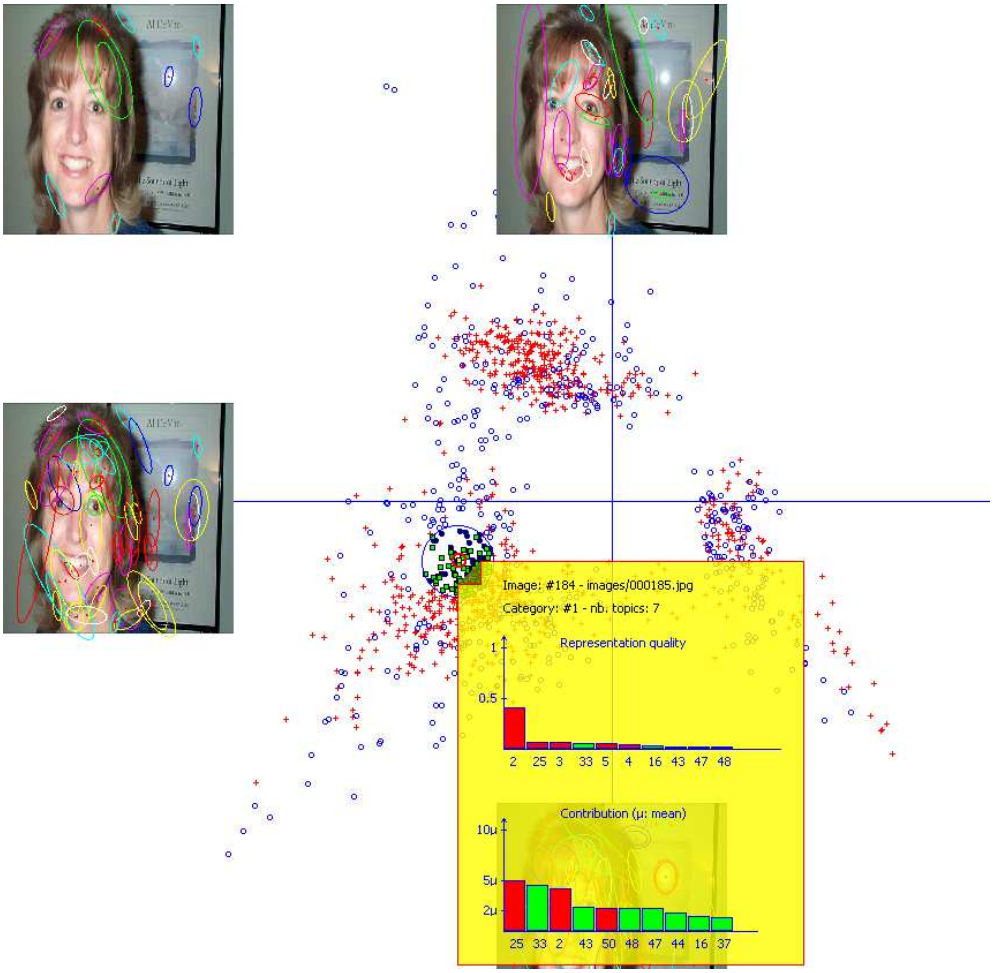


FIGURE 5.6 – Extraction interactive de l'information : qualité de représentation et contribution

(cf. Figures 5.8 et 5.9).

On projette ensuite les points selon ces axes. On trouve alors des thèmes plus fins qui contiennent des images très similaires. En projetant sur les 2 axes 6 et 7, on sélectionne un groupe de points en haut (correspondant à la contribution négative sur l'axe 6 et positive sur le 7) et on voit les images affichées à droite : ces images sont très similaires (ce sont les voitures blanches, cf. Figure 5.10). De même manière, on trouve aussi un autre groupe de points en bas à gauche. En les sélectionnant on trouve un autre thème pour les voitures rouges (cf. Figure 5.11). Enfin, sur les 2 axes 11 et 12, on découvre encore un autre thème pour les voitures (voitures grises, cf. Figure 5.12).

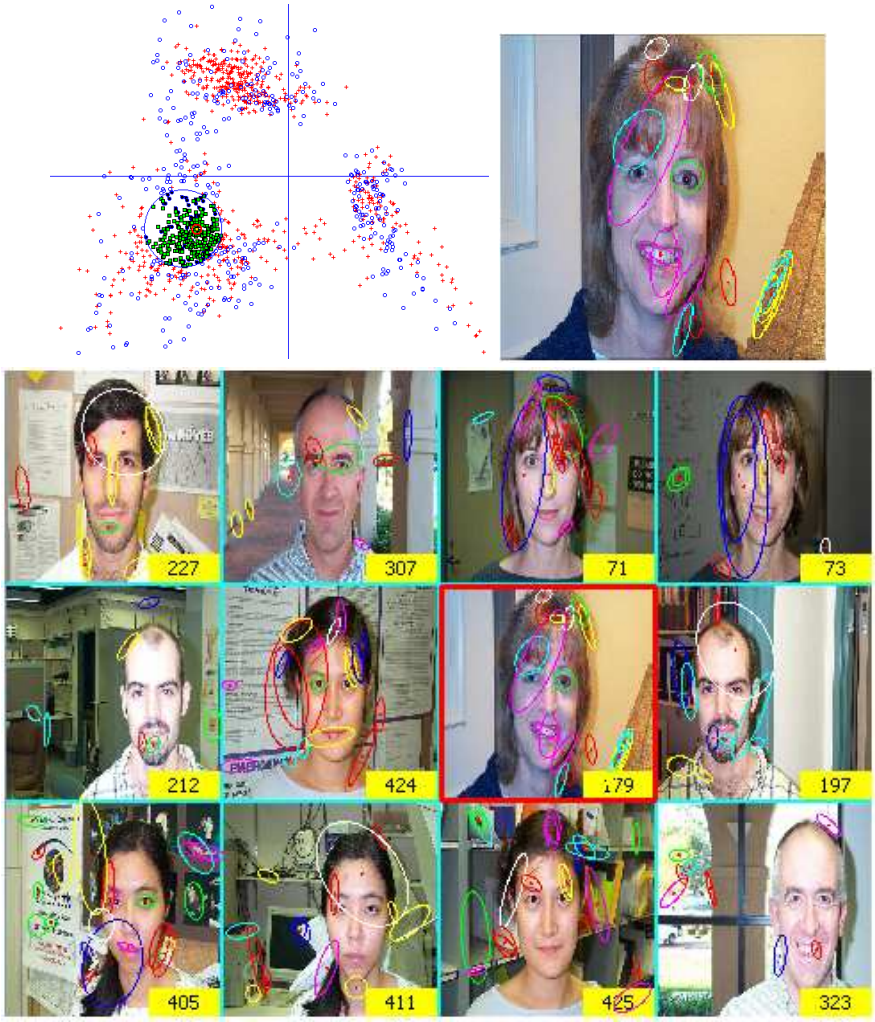


FIGURE 5.7 – Les mots visuels caractérisant le thème « visage »

5.4 Synthèse

Nous avons présenté dans ce chapitre un outil graphique interactif, appelé CAViz, qui permet de visualiser et d'extraire des informations à partir des résultats de l'AFC sur les images afin de mieux comprendre les données et d'interpréter les résultats. Nous présentons aussi une application sur la découverte des thèmes d'images dans la base Caltech-4. Les thèmes obtenus dans cette étude ont montré l'intérêt de CaViz pour l'interprétation de l'AFC en utilisant ses indicateurs comme la qualité de représentation et la contribution aux axes factoriels. Ils nous permettent de découvrir, par exemple, les sous groupes non répertoriés initialement dans la catégorie « voitures » qui correspondent deux différentes couleurs et/ou type des voitures.

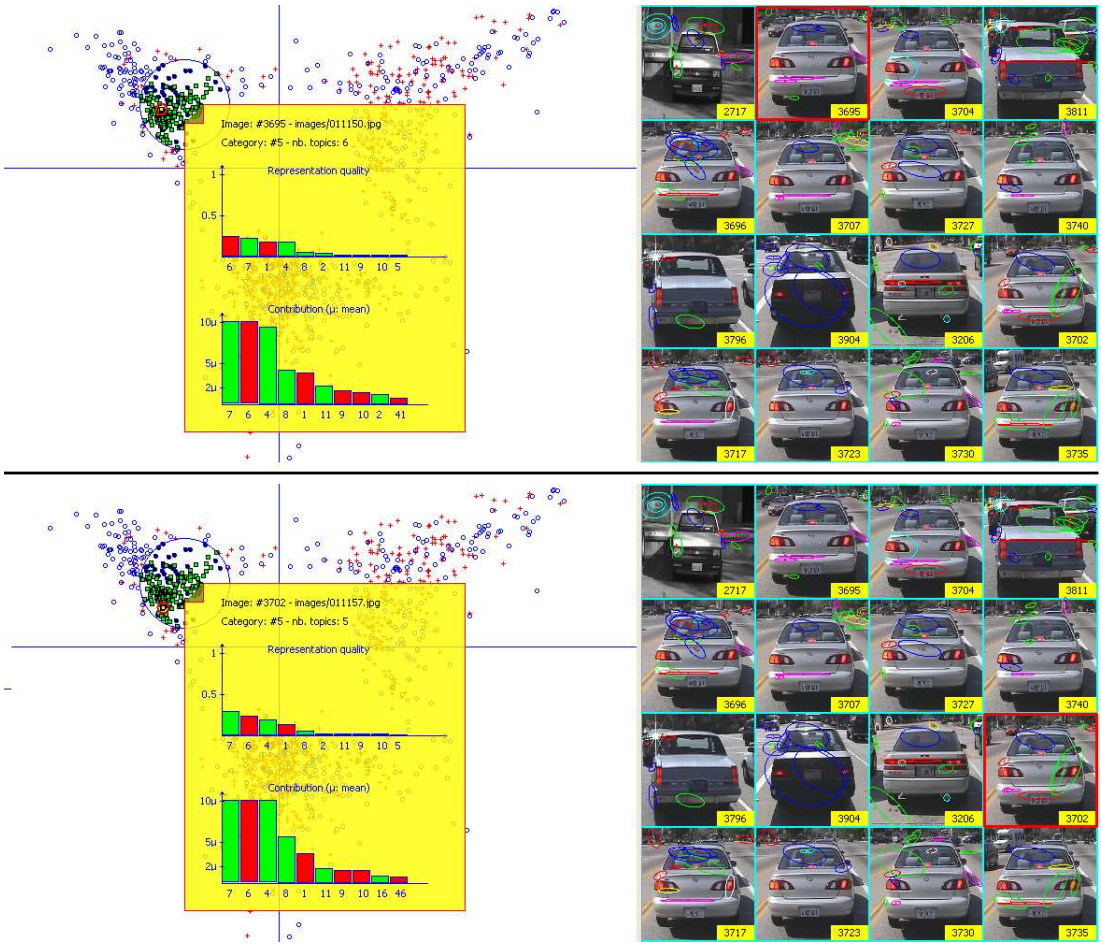


FIGURE 5.8 – Extraction des informations pour trouver de bons axes : deux images bien représentées par les 2 axes 6 (rouge : négatif) et 7 (vert : positif)

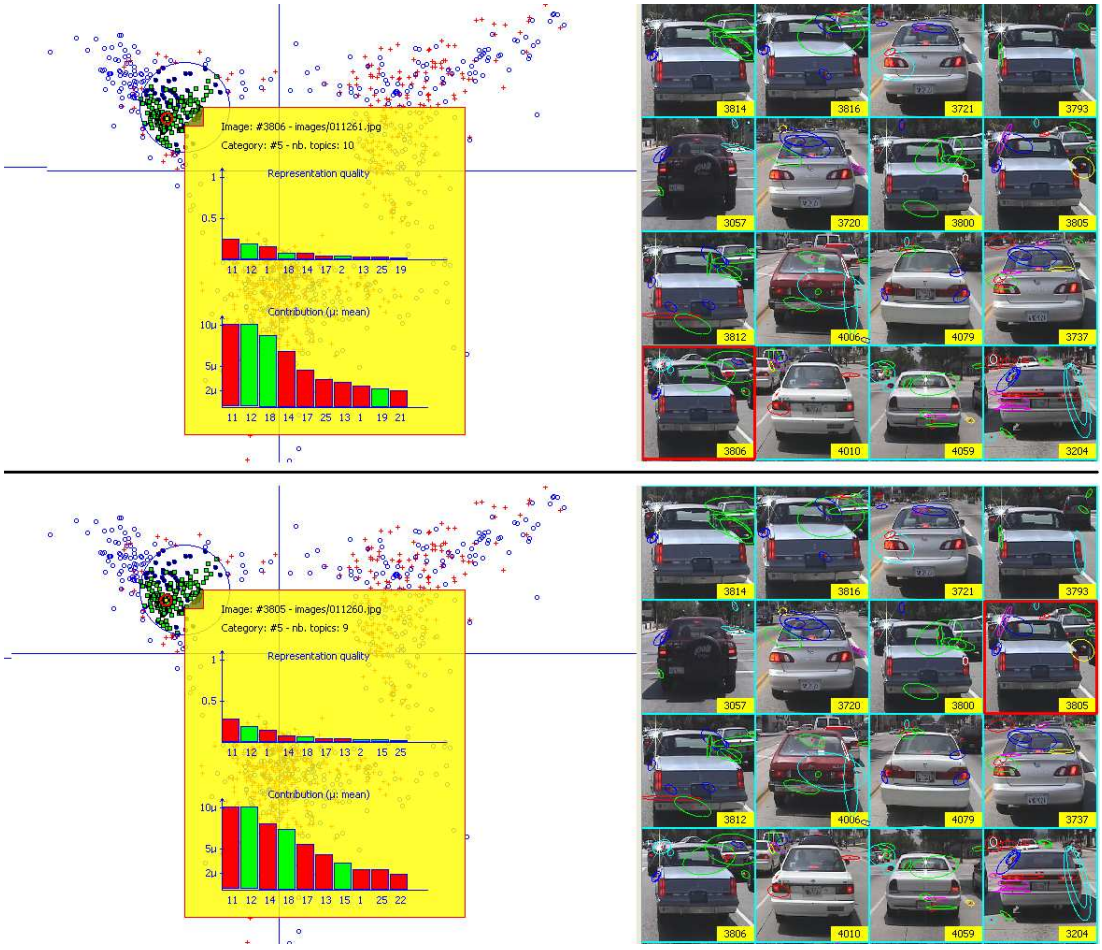


FIGURE 5.9 – Extraction des informations pour trouver de bons axes : deux images bien représentées par les 2 axes 11 et 12

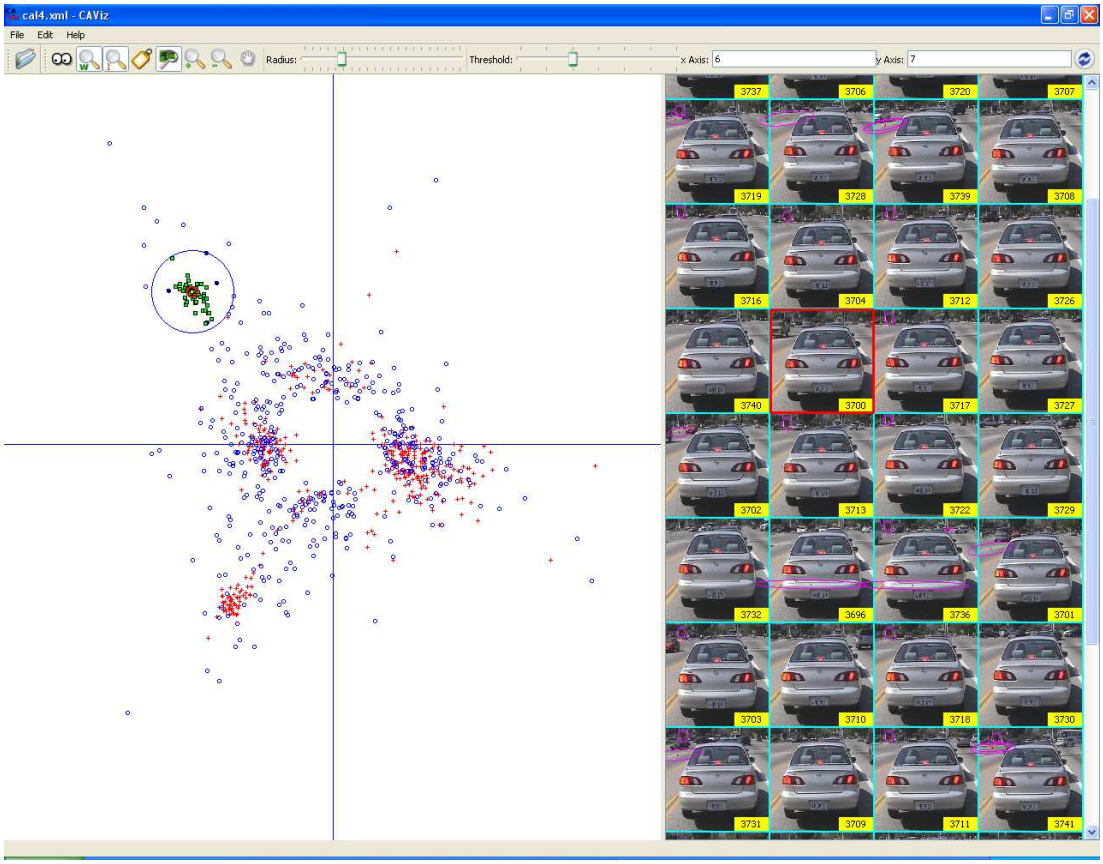


FIGURE 5.10 – Découverte des thèmes en projetant sur le bon plan factoriel : un sous groupe de « voitures » sur le plan 6–7

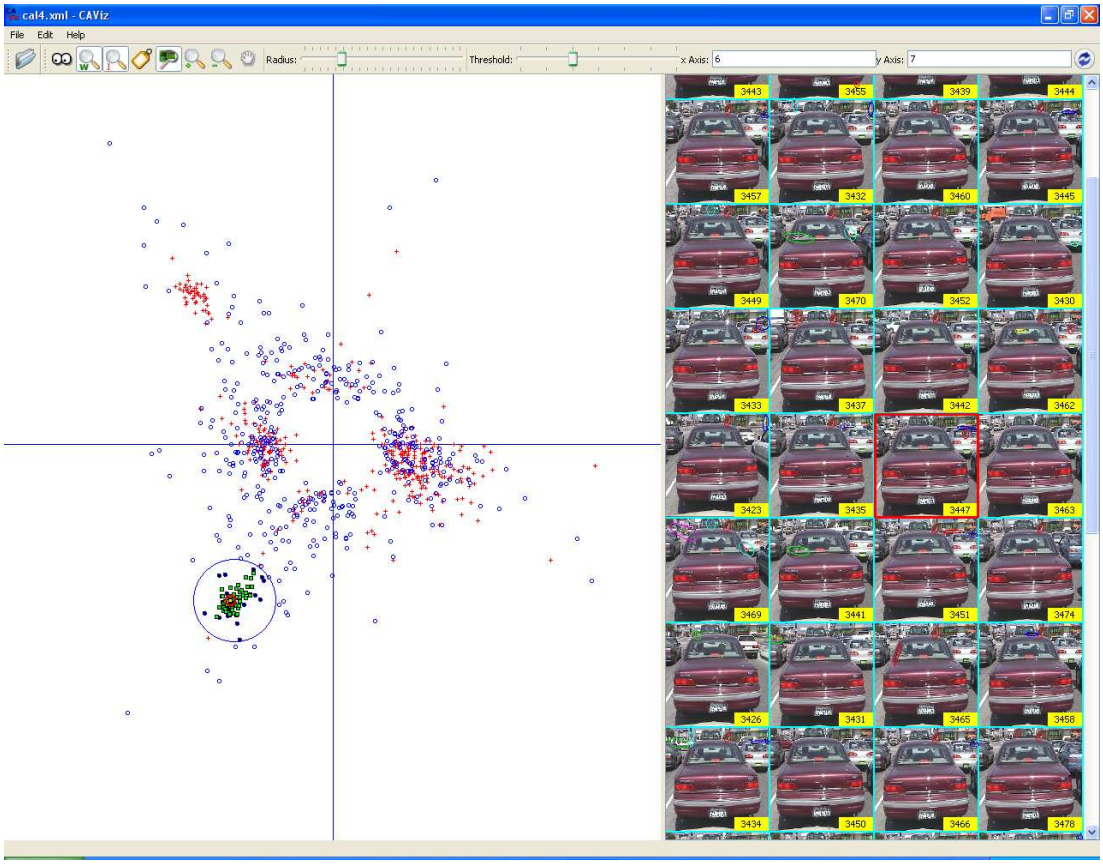


FIGURE 5.11 – Découverte des thèmes en projetant sur le bon plan factoriel : un autre sous groupe de « voitures » sur le plan 6–7

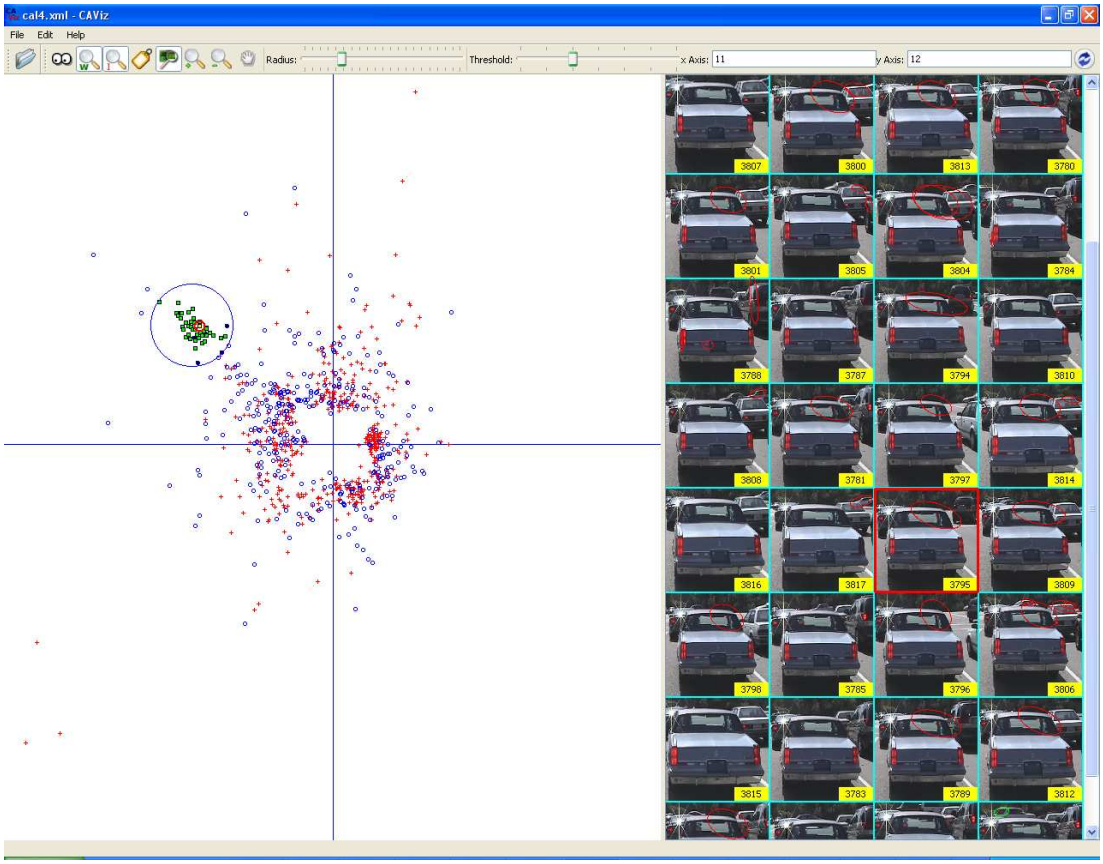


FIGURE 5.12 – Découverte des thèmes en projetant sur le bon plan factoriel : encore un autre thème « voiture » sur le plan 11–12

Conclusion et perspectives

Dans cette thèse, nous nous sommes intéressés à la recherche d'images par le contenu. Nous avons également étudié le problème d'extraction des connaissance à partir d'images, plus particulièrement pour la découverte des thèmes présents dans les images.

Synthèse des travaux effectués

Adaptation de l'Analyse Factorielle des Correspondances aux images. Afin d'adapter l'Analyse Factorielle des Correspondances (AFC) aux images, on construit un tableau de contingence croisant les images et les mots visuels. Ce modèle *sac-de-mots-visuels* est l'analogue de celui utilisé en analyse de données textuelles (ADT) où chaque document est représenté par la fréquence des mots qui s'y trouvent.

Les *mots visuels* sont construits par une quantification vectorielle des descripteurs locaux (par exemple par regroupement à l'aide d'un algorithme de k-means).

Parmi les méthodes inspirées de l'ADT adaptées aux images pour la recherche par le contenu, une pondération simple comme *tf*idf* (term frequency-inverse document frequency) ne donne pas de très bons résultats. D'autres méthodes de modélisation comme le PLSA (Probabilistic Latent Semantic Analysis) et le LDA (Latent Dirichlet Allocation) modélisent les images avec un modèle probabiliste sous-jacent (les paramètres des modèles sont estimés par un algorithme EM) et permettent de les représenter à un niveau sémantique plus élevé (représentation basée sur les thèmes) qu'avec la représentation originale par les fréquences des mots visuels. Les résultats sont donc améliorés par rapport à la pondération *tf*idf*. Cependant, ces méthodes sont très coûteuses en terme de mémoire utilisée et de temps de calcul. Par ailleurs, le nombre de thèmes pour le PLSA et le LDA ne peut pas être très grand car leurs probabilités deviennent trop petites et ne sont plus significatives. De plus, l'algorithme EM ne donne pas toujours une solution globale mais souvent une solution locale.

Nous faisons l'hypothèse qu'une représentation sémantique d'un niveau plus élevé que les fréquences des mots visuels permet d'améliorer la qualité de recherche. Dans un premier temps, nous avons essayé d'adapter l'AFC aux images afin d'obtenir une meilleure représentation des images. Dans ce cas, l'AFC permet de construire des correspondances entre des *thèmes d'images* et les *axes factoriels* de l'AFC. La représentation des images basée sur les axes factoriels (les coordonnées des images sur les axes factoriels) est donc plus sémantique qu'une représentation par la fréquence des mots

visuels. Cela améliore donc les résultats. Par ailleurs, le calcul de l'AFC est très rapide et requiert moins de mémoire que PLSA ou LDA.

Les résultats expérimentaux montrent que l'AFC et le PLSA font beaucoup mieux qu'une pondération *tf*idf* simple. L'AFC améliore encore les résultats par rapport au PLSA.

Accélération de la recherche par des fichiers inversés. La deuxième contribution de cette thèse est une approche originale pour l'accélération de la recherche. Nous avons développé une méthode de recherche approximative basée sur des fichiers inversés qui sont construits à partir des thèmes issus de l'AFC. Il s'agit d'un algorithme de recherche en deux étapes. La première étape consiste à filtrer des images non pertinentes pour une requête donnée. La seconde étape vise à raffiner les résultats par une recherche séquentielle dans une *liste des images candidates* qui restent après l'étape de filtrage.

Pour chaque image, nous déterminons les thèmes auxquels elle appartient. Un thème est associé à une partie d'un axe (négative ou positive). Une image appartient à un thème si elle est bien représentée sur l'axe correspondant et se trouve dans la partie correspondant au thème. Le filtrage des images non pertinentes se base sur le nombre de thèmes que les images partagent avec la requête. Les résultats montrent que notre approche est généralement 10 fois plus rapide qu'une recherche séquentielle sans dégrader la pertinence des images retournées.

Passage à l'échelle et parallélisation. Après avoir adapté l'AFC aux images et obtenu de bons résultats, nous nous sommes intéressés au passage à l'échelle. Le PLSA et le LDA, méthodes se basant sur un algorithme itératif (algorithme EM) exigent, pour chaque itération, que tout le tableau de contingence soit disponible en mémoire. Pour de grands tableaux de contingence, ceci devient impossible.

Nous avons proposé un algorithme d'AFC incrémental pour traiter de très grands tableaux de contingence. Les données sont découpées en bloc par lignes. L'algorithme incrémental traite ces blocs l'un après l'autre et donne les mêmes résultats que la version non incrémentale. Le problème de mémoire est donc totalement résolu.

Par ailleurs, nous avons parallélisé cet algorithme incrémental sur GPU (Graphics Processing Unit). La version parallèle sur 1 GPU est 7 fois (et 10 fois sur 2 GPU) plus rapide que celle non parallèle sur CPU. Le temps de calcul est donc considérablement amélioré.

Nous avons aussi parallélisé l'étape de filtrage de notre algorithme de recherche par des fichiers inversés sur GPU. Sur une base d'un million d'images, l'algorithme parallèle est 10 fois plus rapide que la version non parallèle sur CPU et 100 fois plus rapide qu'une recherche séquentielle sur CPU.

Combinaison avec d'autres méthodes. Nous avons intégré deux mesures de similarité à notre schéma de recherche. Ce sont la Mesure de Dissimilarité Contextuelle (MDC) [JHS07] et la proximité par forêt aléatoire (PFA).

Le principe de la MDC est d'associer à chaque image i, un poids $\delta(i)$ inversement proportionnel à la densité de la région contenant i. Cette pondération améliore la *symétrie de voisinage* d'un schéma de recherche des k plus proches voisins. D'après les auteurs [JHS07], ceci améliore la qualité des résultats. L'inconvénient majeur de la MDC est que le calcul des $\delta(i)$ pour toutes les images dans la base est très coûteux. C'est pour cela que, dans la version originale, ils sont calculés hors-ligne.

En intégrant la MDC à notre schéma de recherche, nous avons utilisé notre méthode de recherche approximative pour calculer efficacement les $\delta(i)$ d'une part et nous avons étendu le calcul de ces poids à la volée, d'autre part. Dans la version à la volée, seuls les $\delta(i)$ des images dans la *liste des images candidates* sont calculés.

En complément de l'AFC, nous avons proposé un algorithme de classification supervisée (en collaboration avec nos collègues) : Forêt aléatoire d'arbres obliques (ou RF-ODT pour Random Forest of Oblique Decision Trees) [DLPP09]. Chaque arbre de décision dans une RF-ODT est construit en partitionnant récursivement les données en deux classes. La recherche du meilleur hyperplan qui sépare les données en deux est effectuée par l'algorithme de Proximal Support Vector Machine (PSVM) [FM01]. La RF-ODT donne de meilleurs résultats (dans le contexte de classification supervisée : les données sont étiquetées) par rapport à l'algorithme SVM standard et la forêt aléatoire proposée par [Bre01].

Nous avons étendu la RF-ODT pour la recherche d'images (dans ce cas, les images ne sont pas étiquetées). On reprend les idées de Breiman sur l'utilisation de la forêt aléatoire dans le cas des données non étiquetées (classification non supervisée) [Bre01].

Concrètement, on génère un ensemble synthétique à partir de l'ensemble à analyser (les images dans l'ensemble original sont étiquetées par +1 et les images dans l'ensemble synthétique par -1) et on construit une forêt aléatoire sur ces deux ensembles. On définit ensuite la *proximité selon une forêt aléatoire* (PFA) entre deux images comme le *nombre de feuilles* où ces deux images se trouvent ensemble.

D'une part, nous avons utilisé directement cette proximité comme une mesure de similarité des images. D'autre part, nous avons exploité la PFA comme procédure de recherche en deux étapes. Dans ce cas, la PFA est considérée comme le nombre de thèmes que les images partagent avec la requête et une feuille de la forêt aléatoire correspond à un thème. Les résultats expérimentaux montrent que la recherche d'images par la PFA est très prometteuse en terme de rapidité et précision.

Visualisation. La dernière contribution de nos travaux de thèse est notre outil graphique interactif, CAViz, en complément des contributions précédentes. CAViz permet de visualiser et interpréter les résultats de l'adaptation de l'AFC aux images. Un des avantages de l'AFC est la représentation conjointe des images et des mots visuels sur un même plan factoriel. Une image est bien caractérisée par des mots visuels qui lui sont associés sur le graphique. CAViz projette des images et mots visuels sur un même plan factoriel. Nous pouvons donc trouver les mots caractérisant une images ou un groupe d'image. Les informations explicatives comme la *qualité de représentation* et la *contribution* sont extraites de manière interactive (avec un couplage de vues). Ceci permet de trouver les axes qui représentent bien une image ou un groupe d'images. La projection

des images sur un bon plan factoriel permet de découvrir des thèmes d'images.

Perspectives

Nous proposons dans cette section plusieurs idées pour poursuivre ce travail. Nous présentons, tout d'abord, quelques extensions de l'analyse factorielle des correspondances (AFC) pour pouvoir traiter des tableaux dont le nombre de lignes et le nombre de colonnes sont simultanément grands. Nous explorons ensuite la possibilité de la fouille d'images à l'aide de l'AFC (éventuellement en combinaison avec d'autres techniques) dans laquelle nous mettons l'accent sur la recherche avec interaction de l'utilisateur et la combinaison entre l'interrogation et la navigation. Nous terminons par quelques voies possibles de l'application de l'AFC sur les images aux problèmes réels.

Extension de l'AFC

Bien que notre algorithme d'AFC incrémental permette de traiter de grands tableaux de contingence (grand nombre d'images ou grand nombre de mots visuels), il faut aussi envisager de traiter des tableaux où le nombre d'images et le nombre de mots visuels sont simultanément grands (par ex., quand on veut utiliser plusieurs mots visuels et/ou combiner des mots visuels et des mots textuels). Dans ce cas, nous suggérons des méthodes de sélection de variables ou des méthodes ensemblistes. Nous avons essayé de faire un *boosting* sur l'AFC [PMG08a]. On a fait une série d'AFC sur des échantillons du tableau de contingence original. Les résultats sont ensuite fusionnés par une analyse en composantes principales pondérées. Les résultats préliminaires sont prometteurs et ferons l'objet de développements futurs.

L'AFC est une méthode d'analyse de données non supervisée (les données ne sont pas étiquetées). Il est intéressant d'introduire les connaissances *a priori* dans l'AFC afin d'améliorer la qualité des résultats. Une autre extension possible est de faire l'AFC locale comme dans le cas de l'ACP locale [KL97].

Fouille d'images à l'aide de l'AFC

Nous présentons, dans cette section, quelques perspectives pour la fouille d'images à l'aide de l'analyse factorielle des correspondances. Nous abordons d'abord les étapes où l'on peut intervenir pour améliorer la qualité du résultat. Nous proposons ensuite une combinaison de l'interrogation et de la navigation qui aide l'utilisateur à explorer et à extraire des connaissances à partir d'une base d'images.

Recherche d'images

Comme la procédure de recherche d'images se compose de plusieurs étapes, le résultat final dépend de ces étapes. Nous pouvons intervenir à chaque étape afin d'améliorer la qualité des résultats.

- ***Construction des mots visuels.*** Le nombre de mots visuels que l'on utilise pour représenter des images joue un rôle important. Un petit k (dans k-means) met ensemble des descripteurs très différents tandis qu'un grand k sépare les variations d'un même descripteur en différents mots visuels. Il s'agit d'une perte d'information à cause de l'étape de quantification. Une solution possible est de garder un nombre de mots visuels assez petit et de faire une affectation floue, c'est à dire qu'un descripteur sera affecté à plusieurs mots visuels. Nous nous intéressons aussi aux méthodes de clustering pour la quantification des descripteurs.
- ***Combinaison des mots visuels et des mots textuels.*** L'utilisation des mots visuels associés à des mots textuels nous intéresse aussi. On pourrait faire les analyses séparément (une pour les mots visuels et l'autre pour les mots textuels) et avant fusion des résultats. On pourrait aussi mettre en ensemble les mots visuels et les mots textuels, faire une seule analyse et trouver des correspondances entre mots visuels et mots textuels.
- ***Intégration des relations spatiales.*** Les relations spatiales entre des descripteurs peuvent être utilisées pour améliorer les résultats. Il est possible (dans l'étape de post-traitement) de vérifier les contraintes géométriques des descripteurs des images dans la liste d'images retournées et de réordonner cette liste.
- ***Interaction homme-machine.*** Le but de recherche d'information est de satisfaire les besoins de l'utilisateur. Seul l'utilisateur peut dire si les résultats retournés sont pertinents pour ses besoins. Les bouclages de pertinence (relevance feedbacks) sont indispensables pour améliorer les résultats.
- ***Répartition de la recherche.*** Nos travaux jusqu'à maintenant traitent le problème de la recherche d'images sur une seule machine. Il faut alors regarder du côté des techniques de répartition des données et des protocoles de communications efficaces entre les machines afin d'adapter les méthodes proposées pour des bases d'images immenses (milliards d'images).

Interrogation et navigation

La plupart des systèmes de recherches d'images existants présentent toujours leurs résultats de recherche par la pertinence des images retournées (par ex., Google et Yahoo !). Ce serait plus intéressant si les images retournées étaient groupées hiérarchiquement selon leurs méta-données et/ou leur contenu visuel. Ceci permettrait de naviguer dans les résultats. Une combinaison de la recherche, de la visualisation et de la navigation permettrait à l'utilisateur d'explorer et d'extraire des connaissances à partir des images.

D'une part, l'application d'une suite d'AFC permet d'organiser les thèmes d'images. La première AFC analyse le tableau de contingence original qui croise les images et les mots visuels. Pour chaque axe factoriel, on obtient deux thèmes qui correspondent aux deux groupes de mots visuels qui contribuent beaucoup à cet axe. On appelle ces groupes *métaclés*. On construit ensuite un autre tableau de contingence qui croise les métaclés obtenues de la première AFC. Chaque élément de ce tableau indique le nombre de mots visuels qui appartiennent à deux métaclés correspondantes. La deuxième AFC effectuée

sur le tableau permet de trouver des thèmes plus fins qui correspondent aux groupes de métaclés. On peut également regrouper les métaclés en "*métamétaclés*" qui servent à la troisième AFC, et ainsi de suite.

D'autre part, la littérature montre que l'Analyse des Concepts Formels (ACF, *Formal Concept Analysis* en anglais) [Wil82] est une bonne méthode à la fois pour l'interrogation et la navigation [GW97]. C'est pour cela que nous nous intéressons à intégrer l'ACF à notre plate-forme de recherche.

En ACF, il y a trois notions principales : *objet*, *attribut* et *concept*. Un *objet* possède certains *attributs*. Un *attribut* est partagé par certains *objets*. Un *concept* est une paire $\langle O, A\rangle$ où O est un ensemble d'objets et A est un ensemble d'attributs. Tous les objets de O doivent posséder tous les attributs de A et un objet quelconque qui possède tous les attributs de A doit être inclus dans O. L'ACF cherche à trouver tous les concepts possibles et les relations entre eux. Les concepts et leurs relations forment un *treillis de concepts*. Ce dernier permet la navigation et l'interrogation dans une base d'objets.

Si on considère les images comme objets et les thèmes issus de l'AFC comme attributs, on pourra appliquer l'ACF sur les images pour obtenir un treillis de concepts. La navigation dans la base deviendra ainsi possible. Une autre combinaison de l'AFC et l'ACF est que, pour chaque requête, on fait l'ACF sur l'ensemble d'images retournées (selon la requête) seulement. Le treillis de concepts obtenu nous permet de naviguer dans les images retournées (pour raffiner les résultats) ou de faire référence au treillis de concepts de la base (dans le cas où nous voulons étendre le rayon de recherche). Par ailleurs, les informations non visuelles associées aux images (méta données) peuvent être utilisées (dans l'AFC et/ou l'ACF) pour améliorer la qualité de la navigation et de la recherche.

Application

Le but de tous les travaux de recherche est l'application des méthodes étudiées aux problèmes réels. Avec les résultats très prometteurs (bonne précision et court temps de réponse) de nos méthodes, il est possible de les appliquer aux problèmes comme la détection de copies ou l'authentification biométrique par images (empreintes, yeux, lèvres) ou de concevoir un moteur de recherche d'images par le contenu à l'aide de l'analyse factorielle des correspondances.

Liste de publications

1. N.-K. Pham, A. Morin, P. Gros, and Q.-T. Le. *Advances in Knowledge Discovery and Management (AKDM'09)*, chapter Intensive use of correspondence analysis for large scale content-based image retrieval (à paraître). 2009.
2. N.-K. Pham, A. Morin, and P. Gros. Accelerating image retrieval using factorial correspondence analysis on GPU. In *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns (CAIP'09)*, volume 5702 of *Springer LNCS*, pages 565 – 572, Münster, Allemagne, 2009.

3. N.-K. Pham, A. Morin, P. Gros, and Q.-T. Le. Utilisation de l'analyse factorielle des correspondances pour la recherche d'images à grande échelle. In *Actes des 9èmes Journées Francophones Extraction et gestion des connaissances (EGC'09)*, volume RNTI-E-15 of *Revue des Nouvelles Technologies de l'Information*, pages 283–294. Cépaduès-Éditions, 2009.
4. N.-K. Pham, A. Morin, and P. Gros. CAViz, an interactive graphical tool for image mining. *Journal of Computing and Information Technology (CIT)*, 16(4) :111–118, 2008.
5. N.-K. Pham, A. Morin, and P. Gros. CAViz, exploration interactive des résultats de l'analyse factorielle des correspondances pour des images. *Revue d'Intelligence Artificielle*, 22(3–4) :489–502, 2008.
6. N.-K. Pham and A. Morin. Une nouvelle approche pour la recherche d'images par le contenu. In *Actes des 8èmes journées Francophones Extraction et gestion des connaissances (EGC'08)*, volume RNTI-E-11 of *Revue des Nouvelles Technologies de l'Information*, pages 475–486, INRIA Sophia Antipolis - Méditerranée, Sophia-Antipolis, France, 2008. Cépaduès-Éditions.
7. N.-K. Pham, A. Morin, and P. Gros. Boosting of factorial correspondence analysis for image retrieval. In *Proceedings of the Sixth International Workshop on Content-Based Multimedia Indexing*, pages 224–229, University of London, Royaume-Uni, juin 2008.
8. N.-K. Pham, A. Morin, and P. Gros. CAViz, an interactive graphical tool for image mining. In *Proceedings of the 30th International Conference on Information Technology Interfaces*, pages 371–376, Cavtat, Croatie, juin 2008.
9. N.-K. Pham, A. Morin, and P. Gros. Recherche d'images par l'analyse factorielle des correspondances. In *Actes de la Conférence en Recherche d'Information et Applications (CORIA'08)*, pages 23–38, Trégastel, France, 2008. Hermès.
10. N.-K. Pham, A. Morin, and P. Gros. Visualization and interactive exploration of factorial correspondence analysis results on images. In *Proceedings of the 3rd International Conference on Human Centered Processes, Part I : Main Conference*, pages 65–80, Delft, Pays-Bas, juin 2008.
11. N.-K. Pham, A. Morin, P. Gros, and Q.-T. Le. Factorial correspondence analysis for image retrieval. In *Proceedings of the IEEE International Conference on Research, Innovation and Vision for the Future (RIVF'08)*, pages 269–275, HCM Ville, Vietnam, 2008.

Glossaire

ACF : Analyse des concepts formels (*Formal concept analysis*)

ADT : Analyse des données textuelles

AFC : Analyse factorielle des correspondances

EM (algorithme) : Expectation maximization algorithm

LDA : Latent Dirichlet allocation

GPU : Graphics Processing Unit

PLSA : Probabilistic latent semantic analysis

PFA : Proximité selon une forêt aléatoire

SIFT : Scale-invariant feature transform

*tf*idf* : term frequency-inverse document frequency

Bibliographie

[ABB+99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[ABP00] J. Assfalg, A. D. Bimbo, and P. Pala. Using multiple examples for content-based image retrieval. In *Proceedings of the International Conference on Multimedia and Expo (ICME'00)*, volume 1, pages 335–338, New York, États-Unis, 2000.

[ACH+91] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3) :209–226, 1991.

[AEK00] M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, (KDD'00)*, pages 179–188, Boston, Massachusetts, États-Unis, 2000.

[AG01] L. Amsaleg and P. Gros. Content-based retrieval using local descriptors : Problems and issues from a database perspective. *Pattern Analysis and Applications, Special Issue on Image Indexation*, 4(2-3) :108–124, 2001.

[ASBH90] K. Arbter, W. E. Snyder, H. Burkhardt, and G. Hirzinger. Application of affine-invariant fourier descriptors to recognition of 3D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7) :640–647, 1990.

[BAG03] S. A. Berrani, L. Amsaleg, and P. Gros. Robust content-based image searches for copyright protection. In *Proceedings of the ACM International Workshop on Multimedia Databases (MMDB '03)*, pages 70–77, La Nouvelle-Orléans, Louisiane, États-Unis, 2003. ACM.

[BBJ+00] S. Berchtold, C. Bøhm, H.V. Jagadish, H.-P. Kriegel, and J. Sander. Independent quantization : an index compression technique for high-dimensional data spaces. In *Proceedings of the 16th International Conference on Data Engineering*, pages 577–588, San Diego, Californie, États-Unis, 2000.

[BBK98] S. Berchtold, C. Böhm, and H.-P. Kriegal. The pyramid-technique : towards breaking the curse of dimensionality. *ACM SIGMOD Record*, 27(2) :142–153, 1998.

[BBV01] N. Boujemaa, S. Boughorbel, and C. Vertan. Soft color signatures for image retrieval by content. In *Proceedings of the 2nd International Conference in Fuzzy Logic and Technology, Leicester, United Kingdom*, pages 394–401, September 2001.

[BCP05] I. Bartolini, P. Ciaccia, and M. Patella. Warp : Accurate retrieval of shapes using phase of fourier descriptors and time warping distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1) :142–147, 2005.

[Bel61] R. Bellman. *Adaptive Control Processes : A Guided Tour*. Princeton University Press, 1961.

[Ben73] J. P. Benzécri. *L'analyse des correspondances*. Paris : Dunod, 1973.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) :509–517, 1975.

[BFSO84] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, janvier 1984.

[BKK96] S. Berchtold, D.A. Keim, and H.-P. Kriegel. The X-tree : An index structure for high-dimensional data. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB '96)*, pages 28–39, Mumbai (Bombay), Inde, 1996. Morgan Kaufmann Publishers Inc.

[BM72] R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indices. *Acta Informatica*, 1 :173–189, 1972.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4) :509–522, avril 2002.

[BNJ03] D. M. Blei, A. Y. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3 :993–1022, janvier 2003.

[Bre01] L. Breiman. Random forests. *Machine Learning*, 45(1) :5–32, 2001.

[BT05] G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 710–715, San Diego, Californie, États-Unis, 2005. IEEE Computer Society.

[BTVG06] H. Bay, T. Tuytelaars, and L. Van Gool. SURF : Speeded-up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV'06)*, pages 404–417, Graz, Autriche, 2006.

[BZM06] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. In *Proceedings of the European Conference on Computer Vision (ECCV'06)*, pages 517–530, Graz, Autriche, 2006.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6) :679–698, 1986.

[CC02] G.-H. Cha and C.-W. Chung. The GC-tree : a high-dimensional index structure for similarity search in image databases. *IEEE Transactions on Multimedia*, 4(2) :235–247, juin 2002.

[CD99] A. Chakraborty and J.S. Duncan. Game-theoretic integration for image segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 21(1) :12–30, 1999.

[CJL88] S. K. Chang, E. Jungert, and Y. Li. Representation and retrieval of symbolic pictures using generalized 2D string. Technical report, University of Pittsburgh, États-Unis, 1988.

[CL01] C. C. Chang and C. J. Lin. LibSVM – a library for support vector machines. Technical report, Department of Computer Science and and Information Engineering, National Taiwan University, Taïwan, 2001.

[Coh91] L.D. Cohen. On active contour models and balloons. *Computer Vision, Graphics, Image Processing : Image Understanding*, 53(2) :211–218, 1991.

[Com79] D. Comer. Ubiquitous B-tree. *ACM Computing Surveys (CSUR)*, 11(2) :121–137, 1979.

[CSY87] S. K. Chang, Q. Y. Shi, and C. W. Yan. Iconic indexing by 2-d strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3) :413–428, 1987.

[CTB+99] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik. Blobworld : a system for region-based image indexing and retrieval. In *Proceedings of the International Conference on Visual Information Systems*, pages 509–516, Amsterdam, Pays-Bas, 1999. Springer.

[CWK05] Y. Chen, J. Z. Wang, and R. Krovetz. CLUE : Cluster-based retrieval of images by unsupervised learning. *IEEE Transactions on Image Processing*, 14(8) :1187–1201, 2005.

[CZZ07] J. Cui, S. Zhou, and S. Zhao. PCR-tree : A compression-based index structure for similarity searching in high-dimensional image databases. In *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'07)*, volume 2, pages 395–400, Haikou, Hainan, Chine, août 2007.

[DDF+90] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harsman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6) :391–407, 1990.

[DJLW08] R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval : Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(2) :1–60, 2008.

[DKNS01] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622, Hong Kong, Chine, 2001. ACM.

[DLPP09] T.-N. Do, S. Lallich, N.-K. Pham, and Lenca P. Un nouvel algorithme de forêts aléatoires d'arbres obliques particulièrement adapté à la classification de données en grandes dimensions. In *Actes de la conférence Extraction et Gestion des Connaissances (EGC'09)*, volume RNTI-E-15 of *Revue des Nouvelles Technologies de l'Information*, pages 79–90, Strasbourg, France, 27-30 janvier 2009. Cépaduès-Editions.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1) :1–38, 1977.

[DMK+01] Y. Deng, B.S. Manjunath, C. Kenney, M.S. Moore, and H. Shin. An efficient color representation for image retrieval. *IEEE Transactions on Image Processing*, 10(1) :140–147, Jan 2001.

[DP04] T.-N. Do and F. Poulet. *Discovery Science*, volume 3245, chapter Enhancing SVM with visualization, pages 183–194. Springer-Verlag, 2004.

[DSH00] Y. Dufournaud, C. Schmid, and R.P. Horaud. Matching images with different resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 612–618, Hilton Head Island, Caroline du Sud, États-Unis, juin 2000. IEEE Computer Society Press.

[dST03] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pages 705–712. MIT Press, 2003.

[DV02] M. N. Do and M. Vetterli. Wavelet-based texture retrieval using generalized gaussian density and Kullback-Leibler distance. *IEEE Transaction on Image Processing*, 11 :146–158, 2002.

[FA91] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9) :891–906, Sept. 1991.

[FFFP04] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples : An incremental bayesian approach tested on 101 object categories. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, volume 12, pages 178–186, Washington, DC, États-Unis, 2004. IEEE Computer Society.

[FGS96] U. Fayyad, Piatetsky-Shapiro G., and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3) :37–54, 1996.

[FGW01] U. Fayyad, G. G. Grinstein, and A. Wierse, editors. *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann Publishers Inc., San Francisco, Californie, États-Unis, 2001.

[FH04] P.F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2) :167–181, 2004.

[FKS03] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD'03)*, pages 301–312, New York, États-Unis, 2003. ACM.

[FM01] G. Fung and O. Mangasarian. Proximal support vector classifiers. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 77–86, San Francisco, Californie, États-Unis, 2001.

[FPZ03] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, volume 2, pages 264–271, Madison, Wisconsin, États-Unis, juin 2003.

[FPZ05] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 380–387, San Diego, Californie, États-Unis, 2005. IEEE Computer Society.

[FSN+95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content : The QBIC system. *IEEE Computer*, 28(9) :23–32, 1995.

[FTAEA00] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi. Vector approximation based indexing for non-uniform high dimensional data sets. In *Proceedings of the ninth international conference on Information and knowledge management (CIKM'00)*, pages 202–209, McLean, Virginie, États-Unis, 2000. ACM.

[GB02] V. Gouet and N. Boujemaa. On the robustness of color points of interest for image retrieval. In *Proceedings of the International Conference on Image Processing (ICIP'02)*, volume 2, pages 377–380, Rochester, New York, États-Unis, 2002.

[GDTG00] L. Germond, M. Dojat, C. Taylor, and C. Garbay. A cooperative framework for segmentation of MRI brain scans. *Artificial Intelligence in Medicine*, 20(1) :77–93, 2000.

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 518–529, Édimbourg, Écosse, Royaume-Uni, 1999. Morgan Kaufmann Publishers Inc.

[GK03] M. Girolami and A. Kabán. On an equivalence between PLSI and LDA. In *Proceedings of the 26th annual international ACM SIGIR Conference on Research and development in informaion retrieval*, pages 433–434, Toronto, Canada, 2003. ACM.

[Gre07] M. J. Greenacre. *Correspondence analysis in practice, Second edition.* Chapman and Hall, 2007.

[GS00] T. Gevers and A.W.M. Smeulders. PicToSeek : combining color and shape invariant features for image retrieval. *IEEE Transactions on Image Processing*, 9(1) :102–119, janvier 2000.

[Gut84] A Guttman. R-trees : A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 47–57, Boston, Massachusetts, États-Unis, 1984. ACM.

[GW97] B. Ganter and R. Wille. *Formal Concept Analysis : Mathematical Foundations.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997. Translator-Franzke, C.

[Har79] R. M. Haralick. Statistical and structural approaches to texture. In *Proceedings of the IEEE*, volume 67, pages 786–804, mai 1979.

[He04] X. He. Incremental semi-supervised subspace learning for image retrieval. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 2–8, New York, États-Unis, 2004. ACM.

[Hea92] D. Heath. *A Geometric Framework for Machine Learning.* PhD thesis, Johns Hopkins University, Baltimore, Maryland, États-Unis, 1992.

[HEMK98] K. Haris, S.N. Efstratiadis, N. Maglaveras, and A.K. Katsaggelos. Hybrid image segmentation using watersheds and fast region merging. *IEEE Transactions on Image Processing*, 7(12) :1684–1699, 1998.

[Hen98] A. Henrich. The LSD^h-tree : An access structure for feature vectors. In *Proceedings of the 14th International Conference on Data Engineering (ICDE'98)*, pages 362–369, Orlando, Floride, États-Unis, 1998. IEEE Computer Society.

[HKM+99] J. Huang, S.R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. Spatial color indexing and applications. *International Journal of Computer Vision*, 35(3) :245–268, 1999.

[HLZ+04] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 9–16, New York, États-Unis, 2004. ACM.

[HMZ04] X. He, W.-Y. Ma, and H.-J. Zhang. Learning an image manifold for retrieval. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 17–23, New York, États-Unis, 2004. ACM.

[Hof99a] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 289–296, Stockholm, Suède, 1999.

[Hof99b] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 50–57, University of California, Berkeley, 1999.

[HS73] R.M. Haralick and K. Shanmugam. Computer classification of reservoir sandstones. *IEEE Transactions on Geoscience Electronics*, 11(4) :171–177, 1973.

[HS79] G.M. Hunter and K.S. Steiglitz. Operations on images using quadtree. *Transactions on Pattern Analysis and Machine Intelligence*, 1(2) :145–153, 1979.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, University of Manchester, Manchester, Royaume-Uni, 1988.

[HS95] G. R. Hjaltason and H. Samet. Ranking in spatial databases. In *Proceedings of the 4th International Symposium on Advances in Spatial Databases*, pages 83–95, Londres, Royaume-Uni, 1995. Springer-Verlag.

[Hu62] M.-K. Hu. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8(2) :179–187, 1962.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors : towards removing the curse of dimensionality. In *Proceedings of the 30th annual ACM symposium on Theory of computing*, pages 604–613, Dallas, Texas, États-Unis, 1998. ACM.

[Jag91] H. V. Jagadish. A retrieval technique for similar shapes. In *Proceeding of the International Conference on Management of Data (SIGMOD)*, pages 208–217, Denver, Colorado, États-Unis, mai 1991.

[JBF05] A. Joly, O. Buisson, and C. Frelicot. Statistical similarity search applied to content-based video copy detection. In *Proceedings of the 21st International Conference on Data Engineering Workshops (ICDEW'05)*, page 1285, Tokyo, Japon, 2005. IEEE Computer Society.

[JF91] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12) :1167–1186, 1991.

[JHS07] H. Jegou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, pages 1–8, Minneapolis, Minnesota, États-Unis, 2007.

[KB02] B. Ko and H. Byun. Integrated region-based image retrieval using region's spatial relationships. In *Proceedings of 16th International Conference on Pattern Recognition (ICPR'02)*, volume 1, page 10196, Québec, Canada, 2002. IEEE Computer Society.

[KBC06] M. Kerbaol, J. Y. Bansard, and J. L. Coatrieux. An analysis of IEEE publications. *IEEE Engineering in Medicine and Biology Magazine*, 25(2) :6–9, 2006.

[KCR09] A. Kerr, D. Campbell, and M. Richards. QR decomposition on GPUs. In *Proceedings of the 2nd Workshop on General Purpose Processing on Graphics Processing Units (GPGPU-2)*, pages 71–78, Washington, DC, États-Unis, 2009. ACM.

[Kei02] Daniel A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1) :1–8, 2002.

[KKOH92] T. Kato, T. Kurita, N. Otsu, and K. Hirata. A sketch retrieval method for full color image database-query by visual example. In *Proceedings of the 11th IAPR International Conference on Patern Recognition, Conference A : Computer Vision and Applications*, volume 1, pages 530–533, Hague, Pays-Bas, 1992.

[KL51] S. Kullback and R. Leiber. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1) :79–86, 1951.

[KL97] N. Kambhatla and T. K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7) :1493–1516, 1997.

[Kle97] J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 599–608, El Paso, Texas, États-Unis, 1997. ACM.

[KS97] N. Katayama and S. Satoh. The SR-tree : An index structure for high-dimensional nearest neighbor queries. In J. Peckham, editor, *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 369–380, Tucson, Arizona, États-Unis, mai 1997. ACM Press.

[KS04] Y. Ke and R. Sukthankar. PCA-SIFT : A more distinctive representation for local image descriptors. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04))*, volume 2, pages 506–513, Washington, DC, États-Unis, 2004.

[KSP95] H. Kauppinen, T. Seppnäen, and M. Pietikäinen. An experimental comparison of autoregressive and fourier-based descriptors in 2D shape classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2) :201–207, 1995.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snake : active contour models. *International Journal of Computer Vision*, 1(4) :312–331, 1988.

[LA04] H Lejsek and F. H. Ásmundsson. The application of the medrank algorithm to content-based image retrieval using local descriptors. Technical report, BS project report, Reykjavík University, 2004.

[LÁJA05] H. Lejsek, F. H. Ásmundsson, B. Th. Jónsson, and L. Amsaleg. Efficient and effective image copyright enforcement. In *Actes des 21èmes journées Bases de Données Avancées (BDA'05)*, Saint Malo, France, 2005.

[LAJA09] H. Lejsek, F. H. Ásmundsson, B. T. Jónsson, and L. Amsaleg. NV-tree : An efficient disk-based index for approximate search in very large high-dimensional collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5) :869–883, Mai 2009.

[LCGMW02] C. Li, E. Chang, H. Garcia-Molina, and G. Wiederhold. Clustering for approximate similarity search in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering*, 14(4) :792–808, 2002.

[LG07] F. Le Guillarm. Encodage et reconstruction d'images grâce à l'analyse des correspondances. Rapport de projet XL, IRISA/ESIEA Laval, février 2007.

[LH90] S.Y. Lee and F.H. Hsu. 2D C-string : a new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10) :1077–1087, 1990.

[Lin98] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2) :79–116, 1998.

[LJF94] K.I. Lin, H. V. Jagadish, and C. Faloutsos. The TV-tree : an index structure for high-dimensional data. *The International Journal on Very Large Data Bases*, 3(4) :517–542, 1994.

[LL00] L.J. Latecki and R. Lakamper. Shape similarity measure based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10) :1185–1190, 2000.

[LMP97] L. Lebart, A. Morineau, and M. Piron. *Statistique exploratoire multidimensionnelle*. Dunod, Paris, 2 edition, 1997.

[Low99] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision*, pages 1150–1157, Corfou, Grèce, 1999.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2) :91–110, 2004.

[LS02] C. A. Lang and A. K. Singh. Accelerating high-dimensional nearest neighbor queries. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM'02)*, pages 109–118, Édimbourg, Écosse, Royaume-Uni, 2002. IEEE Computer Society.

[LS07] R. Lienhart and M. Slaney. PLSA on large scale image databases. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1217–1220, Honolulu, Hawaï, États-Unis, 2007.

[LW04] J. Li and J.Z. Wang. Studying digital imagery of ancient paintings by mixtures of stochastic models. *IEEE Transactions on Image Processing*, 13(3) :340–353, mars 2004.

[LWW00] J. Li, J. Z. Wang, and G. Wiederhold. IRM : Integrated region matching for image retrieval. In *Proceedings of the 8th ACM international conference on Multimedia*, pages 147–156, Marina del Rey, Californie, États-Unis, 2000.

[LYC92] S.Y. Lee, M.C. Yang, and J.W. Chen. 2D B-string : a spatial knowledge representation for image database system. In *Proceedings of the International Computer Science Conference (ICSC'92)*, pages 609–615, Hong Kong, Chine, 1992.

[Mah36] P.C. Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences*, volume 2, pages 49–55, Inde, 1936.

[Mal72] C. L. Mallows. A note on asymptotic joint normality. *The Annals of Mathematical Statistics*, 43(2) :508–515, 1972.

[MGS98] R. Mohr, P. Gros, and C. Schmid. Efficient matching with invariant local descriptors. In *Proceedings of the Joint IAPR International Workshops SSPR'98 and SPR'98 : Advances in Pattern Recognition*, volume 1451 of *Lecture Notes in Computer Science*, pages 54–71, Sydney, Australie, août 1998. Springer-Verlag.

[Mil95] G. A. Miller. WordNet : a lexical database for english. *Communications of the ACM*, 38(11) :39–41, 1995.

[MJ92] J. Mao and A. K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2) :173–188, 1992.

[MKSB93] S. Murthy, S. Kasif, S. Salzberg, and R. Beigel. OC1 : Randomized induction of oblique decision trees. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 322–327, Washington, DC, États-Unis, 1993.

[MM96] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8) :837–842, 1996.

[MM98] W.-Y. Ma and B. S. Manjunath. A texture thesaurus for browsing large aerial photographs. *Journal of the American Society for Information Science*, 49(7) :633–648, 1998.

[Mor04] A. Morin. Intensive use of correspondence analysis for information retrieval. In *Proceedings of the 26th International Conference on Information Technology Interfaces (ITI'04)*, pages 255–258, Cavtat, Croatie, juin 2004.

[MP90] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A*, 7(5) :923–932, 1990.

[MS01] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'01)*, volume 1, pages 525–531, Vancouver, Canada, juillet 2001.

[MS02] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision (ECCV'02)*, pages 128–142, Copenhague, Danemark, 2002. Springer Verlag.

[MS04] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1) :63–86, 2004.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10) :1615–1630, 2005.

[MSB02] J. R. Mathiassen, A. Skavhaug, and K. Bø. Texture similarity measure using Kullback-Leibler divergence between gamma distributions. In *Proceedings of the 7th European Conference on Computer Vision (ECCV'02)*, pages 133–147, Copenhague, Danemark, 2002. Springer-Verlag.

[MTS+05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Internation Journal of Computer Vision*, 65(1-2) :43–72, 2005.

[NBGS08] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with cuda. *Queue*, 6(2) :40–53, 2008.

[NHS84] J. Nievergelt, H. Hinterberger, and K.C. Sevcik. The grid file : An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1) :38–71, 1984.

[NS06] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168, New York, États-Unis, juin 2006.

[NVI08a] NVIDIA. CUBLAS library, version 2.0. `http://developer.download.nvidia.com/compute/cuda/2_0/docs/CUBLAS_Library_2.0.pdf`, 2008.

[NVI08b] NVIDIA. CUDA programming guide, version 2.0. `http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf`, 2008.

[Pap92] T.N. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, 40(4) :901–914, 1992.

[PDM02] E.G.M. Petrakis, A. Diplaros, and E. Milios. Matching and retrieval of distorted and occluded shapes using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11) :1501–1516, novembre 2002.

[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6) :559–572, 1901.

[PF86] E. Persoon and K. S. Fu. Shape discrimination using Fourier descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(3) :388–397, 1986.

[PM08] N.-K. Pham and A. Morin. Une nouvelle approche pour la recherche d'images par le contenu. In *Actes des 8èmes journées Francophones Extraction et gestion des connaissances (EGC'08)*, volume RNTI-E-11 of *Revue des Nouvelles Technologies de l'Information*, pages 475–486, INRIA Sophia Antipolis - Méditerranée, Sophia-Antipolis, France, 2008. Cépaduès-Éditions.

[PMG08a] N.-K. Pham, A. Morin, and P. Gros. Boosting of factorial correspondence analysis for image retrieval. In *Proceedings of the Sixth International Workshop on Content-Based Multimedia Indexing*, pages 224–229, University of London, Royaume-Uni, juin 2008.

[PMG08b] N.-K. Pham, A. Morin, and P. Gros. CAViz, an interactive graphical tool for image mining. In *Proceedings of the 30th International Conference on Information Technology Interfaces*, pages 371–376, Cavtat, Croatie, juin 2008.

[PMG08c] N.-K. Pham, A. Morin, and P. Gros. CAViz, an interactive graphical tool for image mining. *Journal of Computing and Information Technology (CIT)*, 16(4) :111–118, 2008.

[PMG08d] N.-K. Pham, A. Morin, and P. Gros. CAViz, exploration interactive des résultats de l'analyse factorielle des correspondances pour des images. *Revue d'Intelligence Artificielle*, 22(3–4) :489–502, 2008.

[PMG08e] N.-K. Pham, A. Morin, and P. Gros. Recherche d'images par l'analyse factorielle des correspondances. In *Actes de la Conférence en Recherche d'Information et Applications (CORIA'08)*, pages 23–38, Trégastel, France, 2008. Hermès.

[PMG08f] N.-K. Pham, A. Morin, and P. Gros. Visualization and interactive exploration of factorial correspondence analysis results on images. In *Proceedings of the 3rd International Conference on Human Centered Processes, Part I : Main Conference*, pages 65–80, Delft, Pays-Bas, juin 2008.

[PMG09] N.-K. Pham, A. Morin, and P. Gros. Accelerating image retrieval using factorial correspondence analysis on GPU. In *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns (CAIP'09)*, volume 5702 of *Springer LNCS*, pages 565 – 572, Münster, Allemagne, 2009.

[PMGL08] N.-K. Pham, A. Morin, P. Gros, and Q.-T. Le. Factorial correspondence analysis for image retrieval. In *Proceedings of the IEEE International Conference on Research, Innovation and Vision for the Future (RIVF'08)*, pages 269–275, HCM Ville, Vietnam, 2008.

[PMGL09a] N.-K. Pham, A. Morin, P. Gros, and Q.-T. Le. *Advances in Knowledge Discovery and Management (AKDM'09)*, chapter Intensive use of correspondence analysis for large scale content-based image retrieval (à paraître). 2009.

[PMGL09b] N.-K. Pham, A. Morin, P. Gros, and Q.-T. Le. Utilisation de l'analyse factorielle des correspondances pour la recherche d'images à grande échelle. In *Actes des 9èmes Journées Francophones Extraction et gestion des connaissances (EGC'09)*, volume RNTI-E-15 of *Revue des Nouvelles Technologies de l'Information*, pages 283–294. Cépaduès-Éditions, 2009.

[Pou04] F. Poulet. SVM and graphical algorithms : A cooperative approach. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM'04)*, pages 499–502, Brighton, Royaume-Uni, 2004.

[PPS96] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook : Content-based manipulation of image databases. *International journal of computer vision*, 18(3) :233–254, 1996.

[PS00] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1) :49–70, 2000.

[PZ96] G. Pass and R. Zabith. Histogram refinement for content-based image retrieval. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 96–102, 1996.

[PZ99] G. Pass and R. Zabih. Comparing images using joint histograms. *Multimedia Systems*, 7(3) :234–240, 1999.

[Qui93] J. Ross Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, Californie, États-Unis, 1993.

[RJ97] C. Revol and M. Jourlin. A new minimum variance region growing algorithm for image segmentation. *Pattern Recognition Letters*, 18(3) :249–258, 1997.

[RKV95] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 71–79, San José, Californie, États-Unis, 1995. ACM.

[Rob81] J.T. Robinson. The K-D-B-tree : a search structure for large multidimensional dynamic indexes. In *Proceedings of ACM SIGMOD international conference on Management of data*, pages 10–18, Ann Arbor, Michigan, États-Unis, 1981. ACM.

[Ron94] R. Ronfard. Region-based strategies for active contour models. *International Journal of Computer Vision*, 13(2) :229–251, 1994.

[RTG98] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the 6th International Conference on Computer Vision (ICCV'98)*, page 59, Bombay, Inde, 1998. IEEE Computer Society.

[RTG00] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2) :99–121, 2000.

[Sag94] H. Sagan. *Space filling curves*. Springer-Verlag, New York, États-Unis, 1994.

[SB88] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5) :513–523, 1988.

[SB91] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1) :11–32, 1991.

[SC96] J. R. Smith and S. F. Chang. VisualSEEk : A fully automated content-based image query system. In *Proceedings of the ACM International Conference on Multimedia (MULTIMEDIA'96)*, pages 87–98, 1996.

[Sha01] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1) :3–55, 2001.

[SM97] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5) :530–535, 1997.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8) :888–905, août 2000.

[SMB98] C. Schmid, R. Mohrand, and C. Bauckhage. Comparing and evaluating interest points. In *Proceedings of the 6th International Conference on Computer Vision (ICCV'98)*, pages 230–235, Bombay, Inde, 1998. IEEE Computer Society.

[SMB00] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2) :151–172, 2000.

[SO95] M. A. Stricker and M. Orengo. Similarity of color images. In W. Niblack and R. C. Jain, editors, *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases III*, volume 2420 of *SPIE*, pages 381–392, 1995.

[SRE+05] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W.T. Freeman. Discovering objects and their location in images. In *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV'05)*, pages 370–377, Pékin, Chine, 2005.

[SRF87] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-tree : A dynamic index for multi-dimensional objects. In P. M. Stocker, W. Kent, and P. Hammersley, editors, *Proceedings of 13th International Conference on Very Large Data Bases (VLDB'87)*, pages 507–518, Brighton, Royaume-Uni, septembre 1987. Morgan Kaufmann.

[SSH86] M. Stonebraker, T. K. Sellis, and E. N. Hanson. An analysis of rule indexing implementations in data base systems. In *Proceedings of the International Conference on Expert Database Systems*, pages 465–476, 1986.

[SSWC88] P. K. Sahoo, S. Soltani, A.K.C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2) :233–260, 1988.

[SWS+00] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12) :1349–1380, décembre 2000.

[SWY75] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11) :613–620, 1975.

[SZ01] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Proceedings of the 8th International Conference on Computer Vision (ICCV'01)*, pages 636–643, Vancouver, Canada, juillet 2001.

[SZ03a] F. Schaffalitzky and A. Zisserman. Automated location matching in movies. *Computer Vision and Image Understanding*, 92(2–3) :236–264, 2003.

[SZ03b] J. Sivic and A. Zisserman. Video google : A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision (ICCV'03)*, volume 2, pages 1470–1477, Nice, France, octobre 2003.

[SZ06] J. Sivic and A. Zisserman. Video Google : Efficient visual search of videos. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *LNCS*, pages 127–144. Springer, 2006.

[TG99] T. Tuytelaars and L. J. V. Gool. Content-based image retrieval based on local affinely invariant regions. In *Proceedings of the 3rd International Conference on Visual Information and Information Systems (VISUAL'99)*, volume 1614 of *LNCS*, pages 493–500, Amsterdam, Pays-Bas, 1999. Springer-Verlag.

[TMY78] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. *IEEE Transactions On Systems, Man, and Cybernetics*, SMC-8(6), juin 1978.

[TSL+01] Q. Tian, N. Sebe, M. S. Lew, E. Loupias, and T. S. Huang. Image retrieval using wavelet-based salient points. *Journal of Electronic Imaging*, 10(4) :835–849, 2001.

[Uns95] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11) :1549–1560, novembre 1995.

[VFJZ01] A. Vailaya, M. A. T. Figueiredo, A. K. Jain, and H.-J. Zhang. Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10(1) :117–130, 2001.

[VL05] N. Vasconcelos and A. Lippman. A multiresolution manifold distance for invariant image similarity. *IEEE Transactions on Multimedia*, 7(1) :127–142, février 2005.

[WAC+04] J. Willamowski, D. Arregui, G. Csurka, C.R. Dance, and L. Fan. Categorizing nine visual classes using local appearance descriptors. In *Proceedings of the Workshop on Learning for Adaptable Visual Systems, ICPR'04*, Cambridge, Royaume-Uni, août 2004.

[Wan03] Y.-H. Wang. Image indexing and similarity retrieval based on spatial relationship model. *International journal of Information Sciences-Informatics and Computer Science*, 154(1-2) :39–58, 2003.

[Was06] S. Wasson. Nvidia's geforce 8800 graphics processor. Technical report, PC Hardware Explored, 2006.

[WBCST99] W. Wu, K. Bennett, N. Cristianini, and J. Shawe-Taylor. Large margin decision trees for induction and transduction. In *Proceedings of the 6th*

International Conference on Machine Learning (ICML'99), pages 474–483, Bled, Slovénie, juin 1999.

[Wil82] R. Wille. *Ordered sets*, chapter Restructuring lattice theory : an approach based on hierarchies of concepts, pages 445–470. Reidel, Dordrecht-Boston, 1982.

[WJ96] D.A. White and R. Jain. Similarity indexing with the SS-tree. In *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, pages 516–523, La Nouvelle-Orléans, Louisiane, États-Unis, 1996. IEEE Computer Society.

[WLW01] J. Wang, J. Li, and G. Wiederhold. Simplicity : Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 23(9) :947–963, 2001.

[WPD01] R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1–2) :3–35, 2001.

[WSB98] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB'98)*, pages 194–205, New York, États-Unis, 1998. Morgan Kaufmann Publishers Inc.

[YA94] L. Yang and F. Albregtsen. Fast computation of invariant geometric moments : a new method giving correct results. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition - Conference A : Computer Vision & Image Processing*, volume 1, pages 201–204, Jérusalem, Israël, octobre 1994.

[ZWG+04] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[ZY96] S.C. Zhu and A. Yuille. Region competition : unifying snakes, region growing and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9) :884–900, 1996.

[ZZRS00] L. Zhu, A. Zhang, A. Rao, and R. Srihari. Keyblock : an approach for content-based image retrieval. In *Proceedings of the 8th ACM international Conference on Multimedia (MULTIMEDIA'00)*, pages 157–166, Marina del Rey, Californie, États-Unis, 2000. ACM.

Table des figures

Liste des Algorithmes

Résumé

Avec le développement du numérique, le nombre d'images stockées dans les bases de données a beaucoup augmenté. L'indexation des images et la recherche d'information dans les bases d'images sont plus compliquées que dans le cas de documents textuels. Des méthodes d'indexation déjà utilisées en analyse de données textuelles sont proposées pour traiter des images. Pour transférer les résultats de l'analyse de données textuelles aux images, il est nécessaire d'utiliser de nouvelles caractéristiques : les *mots visuels* et on considère les images comme documents.

Nous nous intéressons au problème d'indexation et de recherche d'information dans des grandes bases de données d'images à l'aide de méthodes d'analyse de données, comme l'Analyse Factorielle des Correspondances (AFC). Nous proposons d'abord une utilisation astucieuse des indicateurs de l'AFC pour accélérer la recherche après l'avoir adaptée aux images. Nous nous intéressons ensuite au problème du passage à l'échelle de l'AFC. Pour ce faire, nous proposons un algorithme d'AFC incrémentale pour traiter de grands tableaux de données et la parallélisation de cet algorithme sur processeurs graphiques (GPU). Nous développons aussi une version parallèle de notre algorithme de recherche qui utilise des indicateurs de l'AFC sur GPU. Puis, nous associons l'AFC à d'autres méthodes comme la Mesure de Dissimilarité Contextuelle ou les forêts aléatoires pour améliorer la qualité de la recherche. Enfin, nous présentons un environnement de visualisation, CAViz, pour accompagner les traitements précédents.

Mots-clefs. Analyse factorielle des correspondances, descripteurs locaux, SIFT, indexation, parallélisation, passage à l'échelle, recherche d'images par le contenu, visualisation.

Summary

With the development of the digital world, the number of images stored in databases has significantly increased. Image indexing and information retrieval in image databases are more complicated than in the case of textual documents. Indexing methods already used in textual data analysis are proposed to process images. To transfer the results of the textual data analysis to images, new features are required : *visual words* and images are considered as documents.

We are interested in the problem of indexing and information retrieval in a large database of images using data analysis methods and, more specifically, using Factorial Correspondence Analysis (FCA). First, we propose to use relevant indicators of FCA to speed up the retrieval step after adapting it to images. Next, we study the large scale retrieval with FCA. To this end, we propose an incremental FCA algorithm to deal with large contingency tables, and its parallelization on Graphics Processing Units (GPUs). We also develop a parallel version of our search algorithm using relevant indicators of FCA on GPUs. After that, we combine the use of FCA with other methods such as the Contextual Dissimilarity Measure and random forests in order to improve the retrieval quality. Finally, we present a visualization environment, CAViz, which allows us to display the results.

Keywords. Factorial correspondence analysis, local descriptors, SIFT, indexing, parallelization, large scale retrieval, content-based image retrieval, visualization.